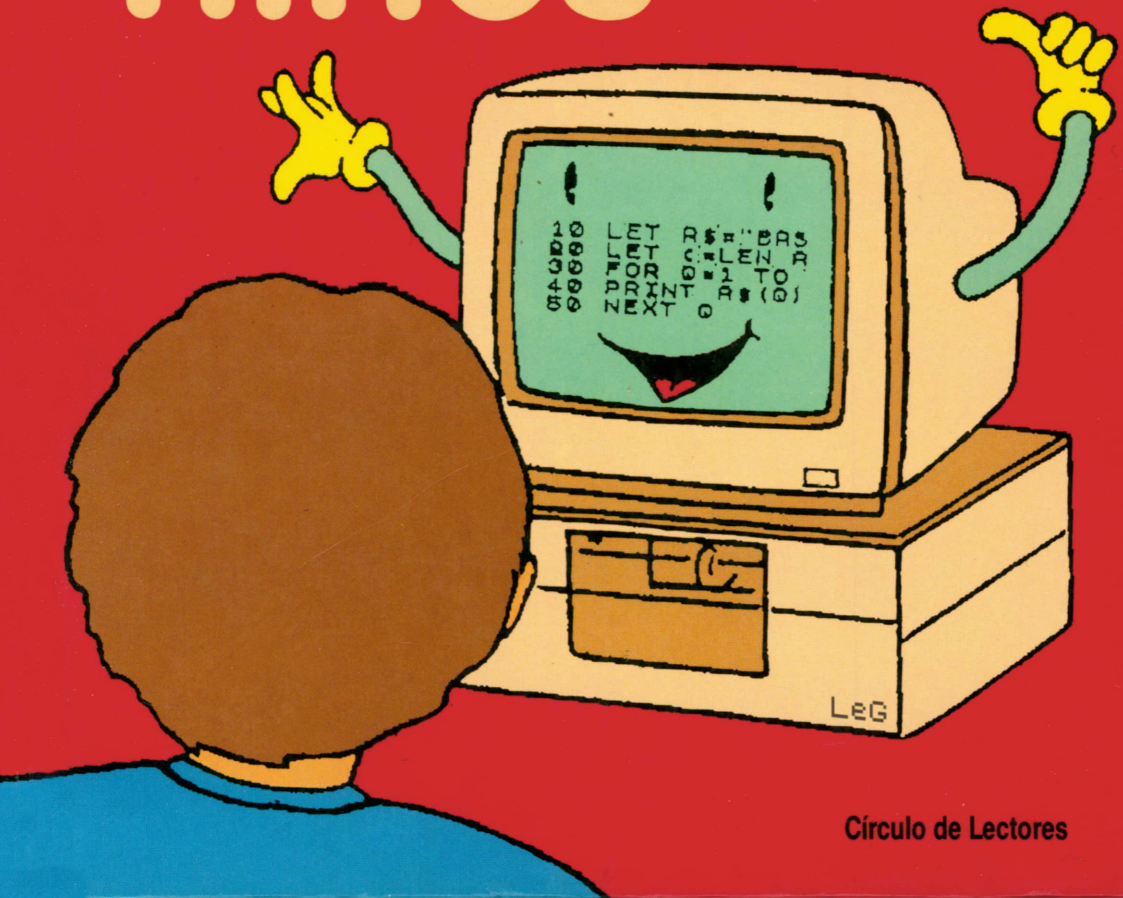
Basic para niños

Sofía Watt

Miguel Mangada



Círculo de Lectores

BASIC para niños

Sofía Watt
Miguel Mangada

BASIC para niños

Círculo de Lectores

A Cristina
porque, ¡... ya está bien!

Indice

BASIC

Unas notas previas 10

Empezamos 17

PRINT, ENTER, RUN, LIST y NEW 23

LET 37

INPUT 49

GO TO 57

IF 65

FOR/NEXT 78

GO SUB 85

READ y DATA 90

REM 98

INT 100

RND 102

Diagramas 113

Juegos 119

BASIC avanzado

Seguimos 129

Mirando dentro 131

La quinta operación 139

Operadores aritméticos 147

CLEAR/CLS 155

INPUT (Algo más) 158

FOR/NEXT (Algo más) 162

Operadores lógicos (AND, OR) 175

IF/THEN (Algo más) (ON GOTO, ON GOSUB, ELSE) 188

READ y DATA 192

RESTORE 197

DIM 200

Ordenando cosas 212

INT 230

Cadenas (LEN, LEFT$, RIGHTS$, MID$, INKEY$, VAL, STR$, GET) 235

CHR$ y ASC 244

Para programar bien 250

Programas 254

Bibliografía consultada 274

BASIC

l
te

es, el
uda, y
decuada-

l forma que
res que carez-

n profundizar más
deden consultar la in-
, publicada en esta mis-

Unas notas previas

En cada uno de los capítulos de BASIC PARA NIÑOS, se estudia un nuevo concepto. Los capítulos se han estructurado en progresión de dificultad conceptual, aunque se reiteran sistemáticamente instrucciones e informaciones ya presentadas, a fin de que se desarrolle adecuadamente la capacidad de asimilación y se mecanicen los conceptos ya vistos.

Este paulatino aumento de información favorece el aprendizaje individual, garantizando la adecuada distribución de energía entre los procesos de recepción y la actividad de tipo elaborador, de forma que niños desde los 8 años y hasta los 14 pueden aprender el lenguaje BASIC permaneciendo en cualquiera de los niveles todo el tiempo que sea necesario hasta que se esté preparado para pasar a un nuevo nivel.

El programa de aprendizaje instrumentalizado en esta obra facilita el refuerzo y afianzamiento de los conceptos fundamentales del lenguaje BASIC. Aun a costa de pecar de repetitivos, se utiliza una permanente "vuelta atrás" sobre lo ya visto, con lo que al final del libro se habrá cubierto el proceso total de aprendizaje: recepción y asimilación, retención y memorización.

Deliberadamente se han excluido determinadas instrucciones en BASIC que, por su complejidad conceptual, escapan de los objetivos de este libro.

Al final de la obra se ofrecen una serie de juegos sencillos, utilizando instrucciones en BASIC ya conocidas.

Unas notas previas

Para lograr con éxito nuestros objetivos, es preciso tener en cuenta una serie de requisitos esenciales:

- El adulto debe leer el texto previamente o con el niño. Conviene también que lea previamente las *notas para el adulto* que se encuentran a lo largo del texto de cada capítulo, antes de trabajar con el niño.

- El niño debe practicar solamente cuando lo desee. En cualquier caso, no debe extenderse en períodos muy largos de tiempo a fin de evitar agotamiento y fatiga mental.

- Es conveniente emplear el número de sesiones que sean necesarias hasta conseguir profundizar en cada concepto.
  No se debe pasar al estudio de un nuevo item si no estamos seguros de haber afianzado profundamente el anterior.

- Hay que permitir al niño que sea él quien avance, quien cree nuevos ejercicios prácticos, nuevas posibilidades. Que descubra por sí mismo.

- Despierte su curiosidad, anímele para que experimente mediante breves indicaciones, si fueran necesarias, tales como "qué pasaría si...", "intenta...". Abra camino a su iniciativa. Los resultados le servirán de estímulo.

Unas notas previas

- No estudien el mismo día más de una instrucción en BASIC. Antes de trabajar hagan un breve repaso de los conceptos fundamentales aprendidos en sesiones anteriores, para comprobar si continúan afianzados.

- Es conveniente que el niño haga con su ordenador todos y cada uno de los programas que se presentan a lo largo del libro, por sencillos que parezcan.

NUESTRO BASIC

Para trabajar con nuestros programas debe utilizar las instrucciones características de su propio ordenador personal. En el libro se hace referencia a un tipo de BASIC convencional y generalizado.

No obstante, puede ocurrir que algunos ordenadores utilicen expresiones diferentes, o distintas combinaciones de teclas para una misma función. Consulte el manual de programación de su propio ordenador cuando tenga alguna duda. Las diferencias serán mínimas. Por ejemplo:

- Algunos ordenadores hacen imprescindible la instrucción **END** al final de cada programa. En otros, ésto no es necesario.

Unas notas previas

- Algunos modelos no utilizan la instrucción **LET**. Por tanto, supriman esta instrucción cuando asignen valores a las variables. Así:

  Escriban 10 A = A + 1

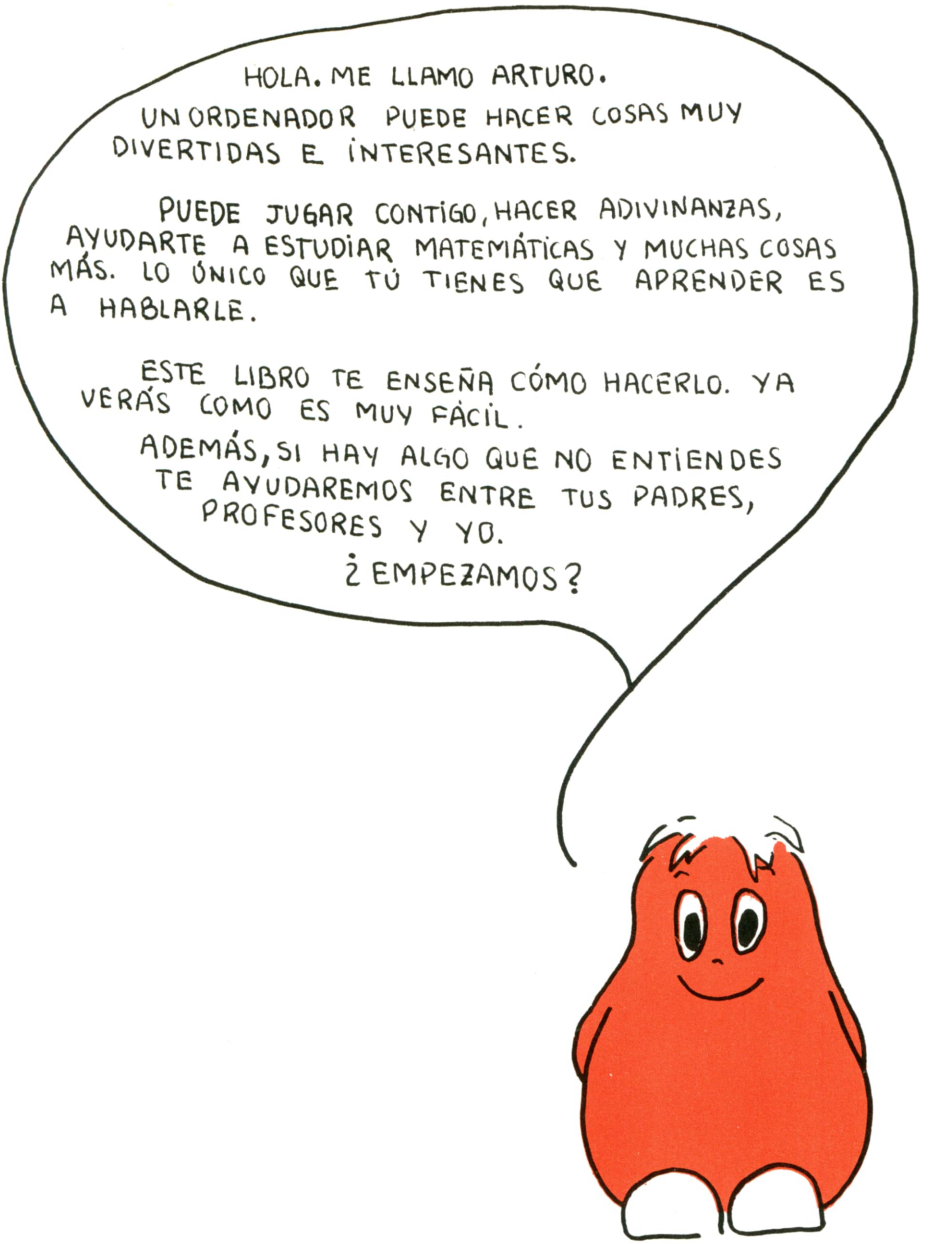
  en lugar de 10 LET A = A + 1

- La función **RND** puede representarse igualmente de distintas formas: es muy frecuente el formato RND (1), RND (0), o RND (X). También se usa, simplemente, RND.

- Algunos ordenadores exigen que se incluya al comienzo del programa la función **RANDOMIZE**, a fin de obtener diferentes series de números aleatorios cada vez que se ejecuta un programa.

- Para interrumpir el programa durante su ejecución, pueden utilizarse distintas instrucciones, dependiendo del tipo de ordenador. Por ejemplo, BREAK o RUN STOP, etc.

- Repetimos que estas diferencias son mínimas y que la lectura del manual de instrucciones de su ordenador aclarará cualquier duda que pueda surgirle.


HOLA. ME LLAMO ARTURO.
UN ORDENADOR PUEDE HACER COSAS MUY DIVERTIDAS E INTERESANTES.
PUEDE JUGAR CONTIGO, HACER ADIVINANZAS, AYUDARTE A ESTUDIAR MATEMÁTICAS Y MUCHAS COSAS MÁS. LO ÚNICO QUE TÚ TIENES QUE APRENDER ES A HABLARLE.
ESTE LIBRO TE ENSEÑA CÓMO HACERLO. YA VERÁS COMO ES MUY FÁCIL.
ADEMÁS, SI HAY ALGO QUE NO ENTIENDES TE AYUDAREMOS ENTRE TUS PADRES, PROFESORES Y YO.
¿EMPEZAMOS?

Empezamos

Como ya te ha dicho ARTURO, un ordenador puede ayudarte en muchas cosas y puedes pasar ratos muy divertidos con él. El ordenador aprende rápido y es también muy rápido en hacer las cosas (muchísimo más rápido que tú), pero para hacerlas necesita que primero seas tú quien le explique lo que tiene que hacer. Y se lo tienes que explicar en un lenguaje que él entienda.

En realidad, el ordenador no sabe hacer casi nada que tú no le hayas enseñado antes. Este libro te explica un idioma o *lenguaje* que tu ordenador entiende, y con el que te puedes comunicar con él. Este lenguaje se llama BASIC.

Imagínate que ARTURO no sabe jugar al escondite. Tú le enseñarías más o menos así: Primero yo cuento hasta 10 y, mientras, tú te escondes. Luego te busco. Si te encuentro, he ganado yo. Si no te encuentro has ganado tú.

Al ordenador le pasa lo mismo que a ARTURO. Tienes que explicarle primero lo que quieres hacer para que él pueda hacerlo.

Pero tienes que tener en cuenta que el ordenador es muy metódico, muy "ordenado", y no le podemos hablar a lo loco. Tenemos que hablarle con mucho orden.

Empezamos

Además tienes que tener siempre en cuenta que el ordenador hará siempre las cosas en el mismo orden en que tú se las digas. Por ejemplo, si le explicaras cómo se juega al escondite, le tendrías que decir:

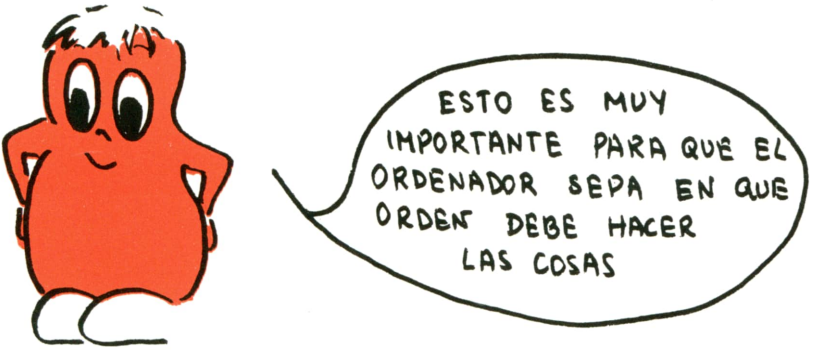
1. Yo cuento hasta diez
2. Tú te escondes
3. Yo te busco
4. Si te encuentro he ganado yo
5. Si no te encuentro has ganado tú.

El ordenador irá *guardando en su memoria,* es decir, recordará la primera línea, después la segunda, luego la tercera, y así hasta el final.

Todo esto lo hace en muy poco tiempo y cuando tú termines de darle las instrucciones de cómo se juega, ya sabrá jugar al escondite. El conjunto de órdenes o *instrucciones* que le has dado, se llama *programa.* Lo que hemos escrito antes sería un programa para jugar al escondite.

Como has podido ver, este programa tiene unas líneas que van numeradas y son las que va leyendo el ordenador para saber cómo se juega. Esas líneas deben ir numeradas, para que él sepa cuál tiene que leer primero, cuál después, etc. Nosotros tenemos que ordenar las líneas siempre desde el número más bajo al más alto, es decir, de menor a mayor.

Empezamos



Escribe en un papel el programa que estamos viendo y efectúa en él los cambios que vamos a ir realizando seguidamente.

Ahora suponte que quieres completar el juego con una nueva regla: Que no vale esconderse dentro de la casa. Entonces escribirías un nueva línea en ese programa que hemos visto antes:

6. Si te escondes en la casa, el juego se acaba

El programa entonces quedaría así:

1. Yo cuento hasta diez
2. Tú te escondes
3. Yo te busco
4. Si te encuentro he ganado yo
5. Si no te encuentro has ganado tú
6. Si te escondes en la casa, el juego se acaba.

Como ves, al ordenador hay que decírselo todo.

Empezamos

Como tenemos que ser muy ordenados, lo mejor hubiera sido colocar esta regla número 6 entre la línea número 2 y la número 3. Así, el juego se parará en cuanto ARTURO haga trampa y se esconda en la casa.

De esta forma nos ahorramos mucho tiempo buscándolo. Fíjate en lo importante que es esto, y la diferencia que hay entre poner la última regla del juego al final o entre la línea 2 y la 3.

Lo que pasa es que no tenemos sitio para meter otra línea entre la 2 y la 3. ¿Qué podemos hacer?

Este truco te va a servir ya para siempre. Es muy sencillo: Numera las líneas no de una en una, sino con la numeración de 10 en 10, porque así siempre tendremos sitio para meter o intercalar, donde nos convenga, otra línea que nos sea necesaria.

Al ordenador le da igual, porque lo único que él necesita es que las líneas estén numeradas de menor a mayor, para saber en qué orden tiene que leer nuestras instrucciones.

Empezamos

De esta forma, el anterior programa quedaría así:

10 Yo cuento hasta diez
20 Tú te escondes
30 Yo te busco
40 Si te encuentro he ganado yo
50 Si no te encuentro has ganado tú.

Y la nueva línea de programa sería cualquiera entre la 20 y la 30 (por ejemplo, la número 25).

25 Si te escondes en la casa, el juego se acaba

Si estuvieras ya dando instrucciones a un ordenador, no te preocuparías por escribir esta línea después de las demás. El ordenador la coloca automáticamente en el sitio que le corresponde.

Cuando consideres que ya no tienes más órdenes que dar y que el programa ya está listo, le darás una orden especial, y el ordenador se pondrá a hacer todo lo que le has enseñado previamente. Esto se llama *ejecutar* el programa o *correr* el programa.

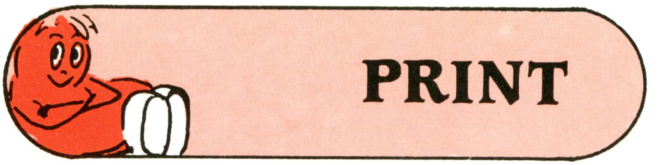
Más adelante veremos la forma de hacer todo esto. Pronto jugaremos al escondite con el ordenador.

Empezamos

NOTAS PARA EL ADULTO

- A través de un lenguaje coloquial y utilizando elementos cotidianos y muy fáciles de asimilar, se habitúa paulatinamente al niño a la terminología específica de la microinformática.

- Repase los conceptos del ejemplo con el niño, utilizando al principio un lenguaje familiar y posteriormente los términos marcados en cursiva en el texto.



PRINT

Como te hemos dicho antes, el ordenador necesita entender solamente unas pocas órdenes e instrucciones muy concretas. Vamos a conocer la primera orden en lenguaje BASIC. Es la orden **PRINT**.

Fundamentalmente, **PRINT** manda al ordenador imprimir todo lo que le indiques y que se encuentre entre los signos “ ” (comillas). El ordenador lo escribirá en la pantalla.

★ Vamos a probar a escribir el nombre de nuestro amigo ARTURO. Utilizando las teclas de tu ordenador, escribe:

```
10 PRINT "ARTURO"
```

Para meter esta línea en la memoria del ordenador, debes pulsar una tecla especial que sirve para eso. Esta tecla tiene distintos nombres según el ordenador que tengas. Puede ser ENTER, RETURN, u otra.

Consulta cuál es.

Fíjate lo importante que es. Cada vez que la aprietas, es como si le dijeras ¡Memorízalo! No olvides teclearla siempre después de escribir cada línea de programa que quieras que el ordenador memorice. Es muy importante, porque de lo contrario olvidará esta línea.

¿Y ahora qué hacemos?

PRINT

Cuando decidas que un programa ya está terminado y no quieres añadir ninguna otra orden, teclea **RUN**.

Esta es la segunda instrucción que vamos a conocer. Sirve para que el ordenador ponga en marcha el programa que tenía en su memoria.

Ya sabes que poner en marcha un programa se llama también ejecutar o correr el programa. La instrucción **RUN** la vas a utilizar siempre que termines un programa y quieras ver qué pasa.

En nuestro caso, como de momento damos por terminado el programa (aunque tenga solamente una línea) vamos a ejecutarlo. Teclea **RUN.**

Verás que en la pantalla aparece el nombre de nuestro amigo.

PRINT

NOTAS PARA EL ADULTO

- Pregunte al niño qué pasaría si hubiera olvidado poner una o las dos comillas en una sentencia **PRINT**. Que experimente los resultados (pueden variar en función del ordenador).
- No olvide recordar al niño que debe pulsar la tecla ENTER O RETURN después de cada orden o instrucción, o para introducir las líneas en el programa.
- El niño debe hacer, con su ordenador, todos y cada uno de los programas que presentamos a lo largo del libro, por sencillos que parezcan.

PROBEMOS OTRAS COSAS

★ Vamos a ampliar el programa haciendo que escriba el nombre de los amigos de ARTURO. Para ello es conveniente ver el programa original.

LIST

Para hacerlo, teclea **LIST.**

¡Ya conocemos otra orden! **LIST** sirve para que podamos ver en pantalla todas las líneas de programa que tiene el ordenador guardadas en su memoria. Es como si le preguntáramos: ¿Qué es lo que ya sabes? Entonces el ordenador *lista* o escribe en la pantalla las líneas de programa que ya teníamos.

Al listar el programa te aparecerá en la pantalla:

```
10 PRINT "ARTURO"
```

Recordarás que éste era nuestro programa inicial. Amplíalo añadiendo otras líneas con los amigos de ARTURO. Teclea:

```
20 PRINT "CARLOS"
30 PRINT "JUANITO"
40 PRINT "JORGE"
```

No te olvides de que después de escribir cada línea, tienes que meterla en la memoria del ordenador (con ENTER o RETURN). Cuando hayas terminado, teclea **RUN** y verás lo que aparece en pantalla. Tienes a toda la pandilla de ARTURO.

NEW

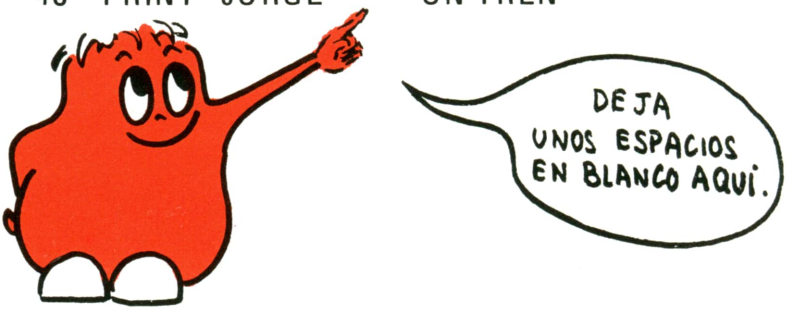
★ ¡Ya has hecho tu primer programa! Vamos a avanzar un poco más haciendo un programa más grande. Como va a ser un programa completamente nuevo, primero borraremos de la memoria del ordenador todo lo que ha aprendido hasta ahora. Para borrar, hay varios sistemas. Ya los conocerás todos. Uno de ellos (muy fácil) consiste en desenchufar el ordenador. Con ello el ordenador vuelve a estar como al principio: Tienes que enseñarle todo otra vez.

Otro sistema (más serio) es teclear **NEW**. Con **NEW** también quitas de la memoria del ordenador todo lo que tenga almacenado o guardado en ella. Ya no se acuerda de nada de lo que le enseñaste antes.

PRINT

★ Hagamos un programa con los amigos de ARTURO y los juguetes que más les gustan. Escribe:

```
10 PRINT "ARTURO      UNA BICICLETA"
20 PRINT "CARLOS      UN BALON"
30 PRINT "JUANITO     UN LIBRO"
40 PRINT "JORGE       UN TREN"
```



Ejecuta el programa (aprieta **RUN**) y verás que aparece en tu pantalla:

```
ARTURO    UNA BICICLETA
CARLOS    UN BALON
JUANITO   UN LIBRO
JORGE     UN TREN
```

★ Y ahora viene algo muy importante. Esto mismo te puede aparecer en la pantalla si cambias el programa de la siguiente manera:

PRINT

```
10  PRINT "ARTURO", "UNA BICICLETA"
20  PRINT "CARLOS", "UN BALON"
30  PRINT "JUANITO", "UN LIBRO"
40  PRINT "JORGE", "UN TREN"
```

Al correr el programa (con **RUN**...) verás que te aparece casi lo mismo de antes. Eso es porque la coma (,) la entiende el ordenador de distinta manera que nosotros. Lo que escribas después de la coma lo escribirá el ordenador más separado de lo anterior.

NOTAS PARA EL ADULTO

- Practique las instrucciones **LIST** y **RUN** hasta que se consiga la soltura necesaria.
- En una instrucción **PRINT**, todos los textos entre comillas separados de otros textos entrecomillados por una coma los imprime el ordenador en la siguiente zona de la pantalla. Esta puede estar dividida en dos o más zonas, según el tipo de ordenador. Presten atención, pues si un texto separado por la coma es demasiado largo, pasará a ocupar la siguiente zona de la pantalla.
- Muestre al niño formas menos "expeditivas" para borrar un programa. Utilicen **NEW.**
- Es imprescindible adecuar el tiempo de estudio dedicado a cada nueva instrucción en lenguaje BASIC a la aptitud y actitud del niño en cada momento.

PRINT

SIGUE PROBANDO

★ A CARLOS le han regalado una raqueta de tenis y ahora ésto es lo que más le gusta. ¿Cómo modificarías el programa?

Fijate que ahora no vas a hacer un nuevo programa, sino a modificar el que ya tienes. Tenemos que cambiar la línea en donde estaba CARLOS. Si no te acuerdas del número de línea, lista el programa y después lo que tienes que hacer es escribir de nuevo esta línea:

```
20 PRINT "CARLOS", "UNA RAQUETA"
```

Mete esta línea en el programa, ejecútalo y fíjate bien en lo que ocurre. Tendrás una pantalla como ésta:

```
ARTURO        UNA BICICLETA
CARLOS        UNA RAQUETA
JUANITO       UN LIBRO
JORGE         UN TREN
```

★ Ahora JUANITO se ha marchado de vacaciones y la pandilla no puede jugar con él. ¿Cómo lo quitarías de tu lista?

PRINT

Escribe solamente el número de línea que corresponde a JUANITO y mételo directamente en la memoria. Es como si le dijeras al ordenador que se acuerde de que esa línea ahora no tiene nada. Al correr el programa verás que JUANITO ya no está en la lista.

★ ¿Quieres formar parte de la pandilla de ARTURO? ¿Cómo incluirías tu nombre en la lista?

Tendrás que añadir una nueva línea de programa, de la misma manera que las veces anteriores.

★ Vamos a probar otra cosa. Teclea:

```
15 PRINT
```

Mételo en la memoria y ejecútalo. ¿Qué ha pasado? Prueba ahora lo que ocurre si añades al programa la línea:

```
25 PRINT
```

Corre el programa y tendrás una pantalla como ésta:

```
ARTURO        UNA BICICLETA
CARLOS        UNA RAQUETA
JORGE         UN TREN
```

PRINT

NOTAS PARA EL ADULTO

- Recuerde al niño que debe listar el programa cada vez que modifique el mismo, para cerciorarse de que lo ha hecho correctamente.
- Para añadir más líneas, utilice números de línea mayores que el de la última línea del programa.
- Para que aparezca una línea en blanco, basta con numerar esa línea, seguida de **PRINT.**
- Haga ver al niño que debe prestar atención para no numerar nuevas líneas con los números de las ya existentes, porque las borraríamos.
- Ayúdele a familiarizarse con las características de su propio teclado, borrando alguna línea o letra erróneamente escrita, utilizando el espaciador, etc.

PRINT

MAS SOBRE PRINT

Con el ordenador también puedes hacer dibujos. Una de las maneras es utilizar la instrucción **PRINT**.

Prueba a hacer un rectángulo con el siguiente programa: Emplea la tecla de los espacios en cuanto lo necesites.

```
10  PRINT "X X X X X X X X"
20  PRINT "X             X"
30  PRINT "X             X"
40  PRINT "X             X"
50  PRINT "X             X"
60  PRINT "X X X X X X X X"
```

Corre el programa y verás lo que pasa.

SIGUE PROBANDO

¿Cómo harías un rectángulo más pequeño?

Borra, por ejemplo, las líneas 40 y 50. Si no recuerdas como se hace, acuerdate de lo que hiciste para quitar a JUANITO de la pandilla de ARTURO. Corre el programa.

PRINT

Si lo has hecho bien, te aparecerá un rectángulo como éste:

```
XXXXXXXX
X      X
X      X
XXXXXXXX
```

Puedes hacer dibujos más completos, combinando distintos símbolos. Prueba a hacer algo parecido a éste:

```
10  PRINT "000$000"
20  PRINT "00$$$00"
30  PRINT "0$$$$$0"
40  PRINT "$$$$$$$"
50  PRINT "0$$$$$0"
60  PRINT "00$$$00"
70  PRINT "000$000"
```

Intenta hacer dibujos más complicados. Prueba con este barquito de vela. Para que se vea mejor, en lugar de rodear el dibujo con un símbolo (como en el caso del rombo, que estaba rodeado por ceros) cuenta los espacios correspondientes al lado izquierdo de la figura y dejalos en blanco.

PRINT

Para ello puedes utilizar la tecla del espaciado. Teclea:

```
 10  PRINT "          X "
 20  PRINT "          XX "
 30  PRINT "          XXX "
 40  PRINT "          XXXX "
 50  PRINT "          XXXXX "
 60  PRINT "          XXXXXX "
 70  PRINT "          XXXXXXX "
 80  PRINT "          XXXXXXXX "
 90  PRINT "          XXXXXXXXX "
100  PRINT "          XXXXXXXXXX "
110  PRINT "          XXXXXXXXXXX "
120  PRINT "          XXXXXXXXXXXX "
130  PRINT "          X "
140  PRINT "          X "
150  PRINT "  XXXXXXXXXXXXXXXXXXXXXXXX "
160  PRINT "   XXXXXXXXXXXXXXXXXXXXXX "
170  PRINT "    XXXXXXXXXXXXXXXXXXXX "
180  PRINT "      XXXXXXXXXXXXXXXXX "
190  END
```

PRINT

NOTAS PARA EL ADULTO

- Para borrar una línea, basta con teclear el número de la línea y pulsar ENTER o RETURN.
- Proponga al niño nuevos dibujos esquemáticos. Sugiérale triángulos, círculos, etc., pero incentive su creatividad animándole a realizar sus propios diseños.
- No olvide teclear **NEW** antes de comenzar un nuevo programa. Esta instrucción borra toda la información contenida en el ordenador. Este comentario es válido siempre que se vaya a comenzar un nuevo programa.
- Conviene que el niño utilice papel milimetrado o cuadriculado para diseñar previamente sus dibujos.

LET

Vamos a conocer ahora la instrucción **LET**. Es muy útil. Para que la entiendas mejor, veámosla con un ejemplo. Fíjate bien en este pequeño programa:

```
10  LET A=14
20  PRINT A
```

★ Corre el programa. Verás que en la pantalla aparece escrito el número 14. ¿Qué ha pasado? La segunda línea del programa ya la entiendes perfectamente. Le has mandado al ordenador que imprima A.

Pero en lugar de escribir A, el ordenador escribe 14. Eso ocurre porque en la primera línea tú le has dicho al ordenador que A vale 14. Eso lo has hecho con **LET**. LET A = 14 es como decir HAZ A = 14.

Fíjate que ahora ya no escribes **PRINT** y luego la letra A entre comillas (“ ”). Eso es porque no quieres que el ordenador te imprima la letra A, sino el valor que tú le has dado a esa letra (en este caso, 14).

LET

★ Vamos a probar otra vez. Escribe el programa:

```
10  LET A=14+1
20  PRINT A
```

Corre el programa y en la pantalla te aparecerá el número 15. Como puedes ver, la instrucción **LET** te permite dar a la letra A el valor que tú quieras. El ordenador guarda en su memoria este valor que tú le has dado. En lugar de la letra A puedes utilizar cualquier otra, o incluso grupos de letras (como AB, HZ, etc.) o combinaciones de letras y números, siempre que empiecen por una letra (como A2, P3, etc.)

Como a todas ellas les puedes dar el valor que tú quieras, se llaman *variables*, porque pueden variar o cambiar su valor.

★ Puedes variar su valor todas las veces que quieras. Por ejemplo:

```
10  LET X=6
20  LET X=8
30  PRINT X
```

LET

Al correr el programa, el ordenador escribe 8, porque en la segunda línea le has dado a la variable X el valor 8, con lo que se anula el valor 6 que le dabas en la primera línea.

★ Fíjate en lo útil que es la instrucción **LET** para hacer cálculos.

Escribe el programa:

```
10  LET M=4
20  LET X=6
30  LET S=M+X
40  PRINT S
```

S ES OTRA VARIABLE. LLAMAMOS S AL RESULTADO DE SUMAR LAS VARIABLES X Y M
NO OLVIDES QUE M Y X TIENEN SUS VALORES Y QUE EL ORDENADOR YA LO SABE.

Cuando corras el programa, el ordenador escribe 10. En las dos primeras líneas has dado a M el valor 4 y a X el valor 6.

En la línea 30 le damos a la variable S el valor de la suma de M + X (que es 10). En la línea 40 ordenamos imprimir ese valor.

LET

NOTAS PARA EL ADULTO

- La instrucción **LET** asigna a una variable situada a la izquierda del signo = el valor de la expresión situada a la derecha de dicho signo.
  Las variables mencionadas en el texto son variables numéricas.

- En algunos ordenadores, no se usa **LET** para asignar valores a las variables. Directamente se escribiría, por ejemplo, 10 X=6 en lugar de 10 LET X=6.

- Haga ver al niño que PRINT A (o cualquier otra variable) hará aparecer en pantalla el valor asignado a esa variable. Comente con el niño la diferencia al escribir PRINT "A".

LET

SIGUE PROBANDO

★ Un programa muy sencillo sería el que nos dice el número total de libros que tiene ARTURO. Tiene 3 del CAPITAN TRUENO, 6 de LOS CINCO y 2 de LOS PITUFOS. Para hacerlo más simple, a la variable T le asignamos los libros del CAPITAN TRUENO. A la variable C, los de LOS CINCO y a la variable P, los PITUFOS.

```
10  LET T=3
20  LET C=6
30  LET P=2
40  LET S=T+C+P
50  PRINT S
```

La variable S será igual a la suma de los distintos tipos de libros que tiene ARTURO. (Mira la línea 40 del programa). Corre el programa para ver el resultado.

★ Si le regalan un nuevo libro de los PITUFOS ¿cómo tendrías que cambiar el programa? Cambia la línea 30 por:

```
30  LET P=3
```

Como ves, con **LET** puedes darle distintos valores a cada variable.

LET

★ La mama de ARTURO le ha dicho que vaya a la tienda a comprar una bolsa de caramelos, que vale 10 pesetas; y dos Pasteles, que valen 25 pesetas cada uno. ¿Cuánto dinero necesitará?

El programa sería:

```
10  LET C=10
20  LET P=25
30  LET X=P*2
40  LET S=C+X
50  PRINT "ARTURO NECESITA ";
60  PRINT S;
70  PRINT "PESETAS"
```

Repasemos con qué variables estamos trabajando y qué valores tiene cada una de ellas:

C: Es la variable que representa los caramelos. Y como sabemos que los caramelos valen 10 pesetas, diremos que C = 10. Por eso escribimos: C = 10.

P: Es la variable que representa a un pastel. Como sabemos que un pastel vale 25 pesetas, diremos por tanto, que P = 25. Por eso escribimos LET P = 25.

LET

X: Es la variable que representa los dos pasteles. Como sabemos que un pastel es P, los dos pasteles serán 2 veces P. Por eso escribimos LET X = P * 2.

S: Es la variable que representa el dinero total que necesita ARTURO. O sea, la suma de lo que cuestan los caramelos más los dos pasteles. Por eso escribimos LET S = C + X.

Corre el programa y verás lo que aparece en pantalla.

★ ¡Una cosa muy importante! Fíjate lo que ha ocurrido cuando ponemos el signo ; (punto y coma) en una instrucción **PRINT**. ¿Qué ha pasado con las siguientes instrucciones **PRINT**? ¿Qué pasaría si quitaras el signo; (punto y coma)?

LET

NOTAS PARA EL ADULTO

- Comente con el niño la línea 30 y su repercusión en el programa.
- El punto y coma (;) en una instrucción **PRINT** hace que la siguiente sentencia **PRINT** se escriba inmediatamente a continuación de la primera. Hágaselo ver al niño.
- Donde corresponda, deberán dejar un espacio en blanco al lado de las comillas, para que las sucesivas sentencias **PRINT** ligadas entre sí a través del punto y coma no se escriban demasiado juntas. Sobre este punto se insiste al niño en próximos capítulos.
- Sugiera al niño que haga otro tipo de cálculos sobre temas relacionados con su entorno familiar. Utilicen sumas, restas, multiplicaciones y divisiones elementales.
- En BASIC, la multiplicación se representa con el signo *. La división, por el signo /.

LET

SIGUE PROBANDO

★ Hasta ahora hemos visto como con la instrucción **LET** le dabas a unas variables el valor que tú querías. Ahora veremos como puedes dar o asignar a una variable incluso palabras o textos.

Sólo tienes que poner detrás de la variable el signo $ y no olvidarte de poner las palabras o textos entre comillas (“ ”).

Teclea el programa:

```
10  LET A$ = "ARTURO"
20  LET J$ = "JORGE"
30  LET C$ = "CARLOS"
```

El ordenador ya tiene en su memoria los nombres que le has dado a cada variable. Compruébalo añadiendo:

```
40  PRINT A$;
50  PRINT " ES AMIGO DE ";
60  PRINT C$
```

LET

Corre el programa. Como ves, el ordenador tiene muy buena memoria. Ya has visto como escribe el ordenador cuando en las sentencias **PRINT** pones un punto y coma (;).

De todas maneras, también puedes obtener el mismo resultado si borras las líneas 40 y 60 y escribes la línea 50 así:

```
50  PRINT A$; " ES AMIGO DE "; C$
```

Corre el programa. Para que te quede bonito, pon mucha atención en la línea 50 (en cualquiera de los dos programas). Debes dejar un espacio en blanco entre las primeras comillas y la palabra ES y también entre la palabra DE y las segundas comillas. Si no lo haces, el ordenador escribirá todo junto.

★ Como ves, resulta muy interesante hacer que todo un grupo de palabras o textos quede asignado a una variable.

LET

Además, puedes trabajar con estas variables casi igual que con las otras. Recordarás por los capítulos anteriores que podías operar con las variables: Las podías sumar, restar, multiplicar, dividir...

Estas variables que acabamos de ver ahora solamente se pueden sumar. No tiene sentido que digas "ARTURO" multiplicado por "CARLOS". Sin embargo, sí puedes decir "ARTURO" y "CARLOS".

Vamos a ver un programa que nos sirva de ejemplo.

```
10  LET A$="ARTURO "
20  LET B$="ES "
30  LET C$="NO ES "
40  LET D$="ALTO"
50  LET E$="LISTO"
60  LET F$="FEO"
70  LET G$="BUENO"
80  PRINT A$+B$+D$
```

Corre el programa. Después, cambia la línea 80 poniendo distintas variables. Por ejemplo, PRINT A$ + C$ + F$. O también, PRINT A$ + B$ + G$.

Como ves, se pueden sumar estas variables y, por tanto, los grupos de textos que les corresponden.

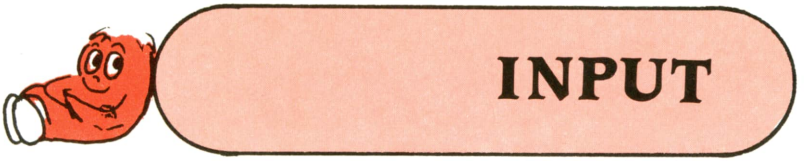
LET

NOTAS PARA EL ADULTO

- Estas variables se denominan variables alfanuméricas, pues a ellas se asignan cadenas de caracteres numéricos, alfabéticos o alfanuméricos. Unicamente pueden ser sumadas y entre sí. No puede sumarse una variable numérica y una alfanumérica.

- En el texto anterior se ha utilizado la expresión "textos" o "grupos de palabras" al hacer referencia a las cadenas de caracteres. Utilice inicialmente estas expresiones, para sustituirlas paulatinamente por una terminología más ortodoxa, denominando preferentemente a éstas "cadenas de caracteres".

- Remarque nuevamente la atención del niño sobre la función de las comillas, y la colocación de los espacios en blanco dentro de los textos incluidos en una sentencia PRINT. Observe que en las sentencias PRINT ligadas entre sí por un punto y coma (;) es conveniente dejar espacios en blanco entre el texto y las comillas, para evitar que se junten textos y/o variables alfanuméricas.



INPUT

Esta instrucción es una de las más bonitas. Con ella puedes "hablar" con el ordenador, que te pregunta y te responde.

INPUT te permite introducir datos con tu teclado mientras se está ejecutando o corriendo un programa. Fíjate en el ejemplo:

```
10  LET A=2
15  PRINT " ESTOY ESPERANDO UN NUMERO"
20  INPUT M
30  LET B = A * M
40  PRINT B
```

★ Corre el programa. El ordenador empieza a ejecutarlo. Al leer la primera línea, asigna el valor 2 a la variable A. Entonces llega a la instrucción **INPUT** y lo que hace es pararse esperando un número que tú le tienes que dar a través del teclado.

INPUT

Este dato es el valor que tú le vas a dar a la variable M. Es decir, el ordenador se dice a sí mismo: "Estoy esperando un dato, y el valor de ese dato se lo voy a asignar a la variable M". Esto es lo que hace **INPUT**.

Dale a M el valor que tú quieras. Por ejemplo, 5. Para ello, pulsa 5 (y luego ENTER o RETURN). Entonces el ordenador sigue con el programa y como ya todas las variables tienen su correspondiente valor, hace las operaciones que le has indicado en la línea 30 e imprime el resultado.

En nuestro ejemplo toma A que vale 2 y M al que le has dado el valor de 5 y los multiplica imprimiendo el resultado: 10.

★ Prueba dando distintos valores a M y luego cambiando el valor de A.

NOTAS PARA EL ADULTO

- La sentencia **INPUT** permite la interacción entre el ordenador y el usuario. Posibilita la entrada de datos en el transcurso de la ejecución de un programa. Combinando instrucciones **PRINT** e **INPUT** podremos conseguir una relación "conversacional" entre el niño y el ordenador, como veremos más adelante.

INPUT

SIGUE PROBANDO

★ Vamos a hacer un programa para que puedas hablar con tu ordenador. Escribe:

```
 10  PRINT "ESCRIBE TU NOMBRE"
 20  INPUT N$
 30  PRINT "¿EN QUE AÑO HAS NACIDO ";N$;" ?"
 40  INPUT A
 50  PRINT "¿EN QUE AÑO VIVIMOS?"
 60  INPUT B
 70  LET X=B-A
 80  PRINT
 90  PRINT
100  PRINT
110  PRINT
120  PRINT
130  PRINT"BUENO ";N$;" TIENES UNOS ";X;
     " AÑOS"
```

Corre el programa y contesta las preguntas. ¡El ordenador puede hablar contigo!

INPUT

Vamos a analizar un poco el programa. En las líneas de programa 10, 30 y 50 estás enseñando al ordenador a preguntarte. En la línea 20 verás que ponemos el signo $ detrás de la variable. Eso es porque a esta variable no le vas a asignar un valor numérico sino una palabra. Si no te acuerdas muy bien de ésto, vuelve a leerte la página 45.

En las líneas 40 y 60 no ponemos el signo $ porque los valores de A y B van a ser numéricos. Puedes ver en el programa que A es el año en que has nacido y B el año en que vives.

Con las líneas 80 a 120 hemos dejado líneas sin escribir para que la respuesta quede más separada de las preguntas que has hecho antes.

Vamos a hacer una modificación muy interesante. Hagamos que nuestras respuestas queden también escritas en la pantalla. Añade las líneas:

```
25  PRINT N$
45  PRINT A
65  PRINT B
```

Corre de nuevo el programa y observa las diferencias.

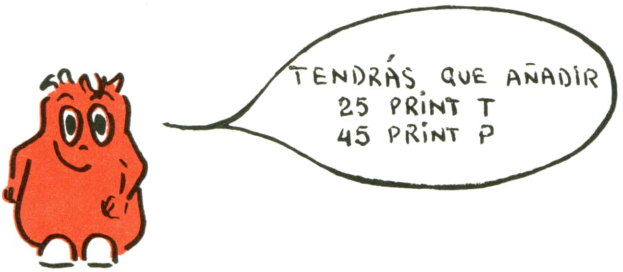
INPUT

★ ARTURO está preparando sus vacaciones y quiere saber cuantos días va a pasar en su casa. Como ya sabe cuantos días va a estar en la playa, haría el siguiente programa:

```
10  PRINT "¿CUANTOS DIAS DE VACACIONES
    TENGO?"
20  INPUT T
30  PRINT "¿CUANTOS DIAS ESTARE EN LA
    PLAYA?"
40  INPUT P
50  LET D=T-P
60  PRINT "ME QUEDAN ";D;" DIAS DE VACA-
    CIONES EN CASA"
```

Ejecuta el programa. El ordenador te irá preguntando. Contesta tú por ARTURO. El ordenador te dará la respuesta final.

Al igual que hiciste en el programa anterior, modifica éste, de forma tal que aparezcan tus respuestas en la pantalla.



INPUT

★ A ARTURO le gusta mucho la playa y quiere saber cuántas horas va a poder disfrutar de la arena y del mar. Para hacer el cálculo, utiliza el siguiente programa. Tú ya sabes que un día tiene 24 horas.

```
10 PRINT "¿CUANTOS DIAS PASARA ARTURO EN
   LA PLAYA?"
20 INPUT P
30 LET H=24
40 LET S=P*H
50 PRINT "ARTURO PASARA ";S;" HORAS EN LA
   PLAYA"
```

Corre el programa y tendrás la respuesta. Modifícalo para que aparezca tu respuesta en pantalla.

Sin embargo, ARTURO no se da cuenta de que no se puede pasar todas las horas del día en la playa. Tiene que dedicar algunas horas a descansar. Mediante la siguiente ampliación del programa ayúdale tú a ver cuántas horas le quedan de verdad para jugar.

Contesta tú mismo a las preguntas que te haga el ordenador cuando se esté ejecutando el programa completo.

INPUT

Añade primero al programa anterior:

```
55  PRINT
56  PRINT
57  PRINT
58  PRINT
60  PRINT "DIME LAS HORAS QUE
    DESCANSA ARTURO CADA DIA"
70  INPUT X
75  PRINT X
80  LET Z=24 - X
90  LET Q=Z * P
100 PRINT "ARTURO JUGARA EN LA PLAYA  ";
    Q ;" HORAS DURANTE SUS VACACIONES"
```

Si además quieres ayudarle a que sepa cuántas horas jugará cada día, cambia la línea 100 ampliándola así:

```
100 PRINT "ARTURO JUGARA EN LA PLAYA  "; Q;
    "  HORAS DURANTE SUS VACACIONES, O SEA ";
    Z;" HORAS CADA DIA"
```

INPUT

NOTAS PARA EL ADULTO

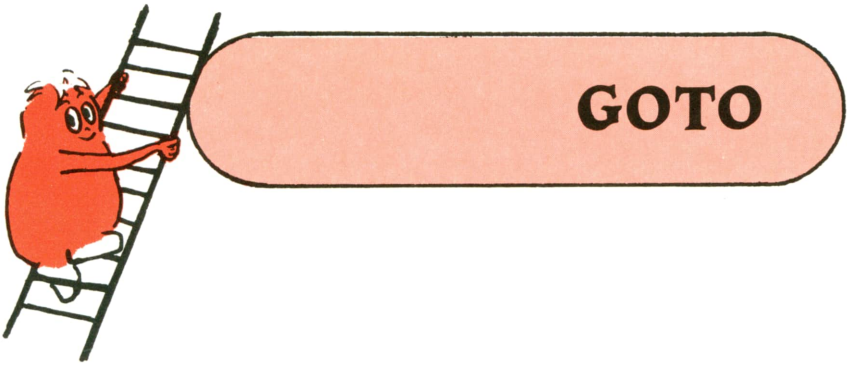
- Trabaje insistentemente con este programa. Sugiera al niño la utilización de **LIST** cada vez que sea conveniente. Sean cuidadosos en la colocación de los espacios en blanco entre las sentencias **PRINT**, a fin de que las frases queden correctamente escritas.

- Reitere la importancia del punto y coma. Signifíquele al niño la diferencia de utilizar la coma a modo de instrucción, para escribir en la siguiente zona de la pantalla o de incluirla dentro de una instrucción **PRINT**, como en la línea 100. Practique algunos ejemplos de su invención situando la coma dentro y fuera de la parte entrecomillada de una sentencia **PRINT.**

- Repase varias veces el programa, analizando con el niño los distintos valores asignados paulatinamente a las diferentes variables a través de la sentencia **LET.**

- Hágale ver la forma y los números de línea en los que se han efectuado añadidos y modificaciones.

- Sobre esta base, motive al niño a inventar nuevas situaciones y desarrollen conjuntamente los programas correspondientes.



GOTO

Vamos a hacer nuestros programas más completos, utilizando **GO TO**.

Con esta instrucción le das una orden tajante al ordenador. Significa VETE A, y el ordenador tiene que ir a la línea de programa donde tú le mandes.

Esta instrucción te alterará el orden normal de ejecución del programa.

★ Por ejemplo, en este programa:

```
10  PRINT "HOLA"
20  GO TO 10
```

El ordenador lee la primera línea y escribe HOLA. Luego lee la segunda línea, que le ordena ir a la línea 10. Entonces imprime otra vez HOLA. Luego lee la segunda línea, que le ordena ir a la línea 10. Entonces imprime otra vez HOLA y así sucesivamente. ¡Este programa no termina nunca! Córrelo y verás qué pasa.

GOTO

En tu pantalla hay una columna de palabras HOLA. Aunque ya no tenga sitio para seguir escribiendo, el ordenador sigue por dentro ejecutando el programa. Para que pueda pararse, tienes que teclear BREAK, que es una instrucción que interrumpe los programas mientras se están ejecutando. (Tal vez tú ordenador utiliza otra instrucción en lugar de BREAK. Consulta cuál es.)

Cambia la línea 10 del programa por:

```
10 PRINT "HOLA",
```

y comprobarás algo que ya conoces sobre la importancia de la coma.

Lista el programa (no olvides pulsar antes BREAK).

Cambia la línea 10 otra vez, poniendo un punto y coma (;) en lugar de la coma. Recordarás así la influencia del punto y coma.

No te olvides que no es lo mismo poner una coma dentro de las comillas (“ ”) o fuera de ellas. Prueba poniendo la línea 10 del programa así:

```
10 PRINT "HOLA,"
```

Volviendo a la instrucción **GO TO**, vamos a ver algunos programas donde veremos la utilidad que tiene.

GOTO

★ Por ejemplo, ARTURO quiere contar números de dos en dos. Podríamos escribir el programa:

```
10  LET A=0
20  PRINT A
30  LET A=A+2
40  PRINT A
50  LET A=A+2
60  PRINT A
```

Este programa sólo cuenta hasta 4. Si queremos que siga contando, el programa será larguísimo...

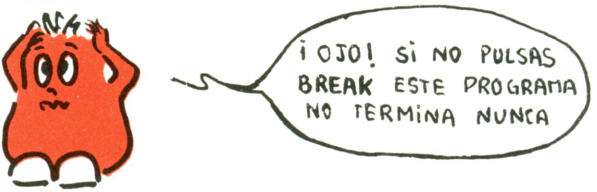
Vamos a utilizar **GO TO**. Escribe el siguiente programa:

```
10  LET A=0
20  LET A=A+2
30  PRINT A
40  GO TO 20
```

GOTO

Al correr el programa te aparecerá en la pantalla lo mismo que la vez anterior:

2
4
6



Fíjate bien en el programa. En la línea 10, le has dado un valor a la variable A. En este caso, el valor 0.

En la línea 20, le das un nuevo calor a A: Su valor inicial más las unidades en que quieres aumentar dicho valor (en nuestro ejemplo, dos unidades). Es decir, A vale ahora 2.

En la línea 30, imprimes el valor que tiene A en ese momento (es decir, 2).

En la línea 40 le ordenas que vaya a la línea 20 y entonces el ordenador toma el último valor de A (que era en este momento 2) y le suma 2 unidades. Entonces A vale ahora 4.

A continuación imprimirá este valor y, obligado otra vez por la instrucción **GO TO** de la línea 40, irá de nuevo a la línea 20, donde tomará el último valor de A (que ahora es 4) y le suma dos unidades. Imprime el nuevo valor (que ya es 6) y así sucesivamente.

Conviene que sigas estas explicaciones con el programa a la vista.

GOTO

★ Haz tú mismo un programa para contar de 5 en 5. Para ello, cambia la línea 20. Si no lo sabes, fíjate en lo que dice ARTURO.

NOTAS PARA EL ADULTO

- La instrucción **GO TO** implica un salto o transferencia incondicional. Altera el orden secuencial de las líneas de programa. Estos saltos pueden ser hacia líneas de mayor o menor número que aquella en la que se encuentra la instrucción **GO TO.**

- Haga ver al niño que en los programas vistos en este capítulo, cada vez que a través de la instrucción **GO TO** el ordenador vuelve a una línea de programa anterior, ya tiene en memoria los últimos valores que ha adoptado la variable.

- Fíjese que en matemáticas la expresión A = A + 2 no tiene sentido. En informática sí, ya que primero se calcula el valor situado a la derecha del signo = y después se asigna ese valor a la variable que está a la izquierda de dicho signo.

GOTO

SEGUIMOS PROBANDO

★ ARTURO está aprendiendo a sumar y quiere hacerse una tabla de sumas. Vamos a ayudarle haciendo un programa que le sirva para, por ejemplo, obtener y aprender a sumar añadiendo 3 unidades a cualquier número natural. Teclea:

```
10  LET X=0
20  LET S=X+3
30  PRINT X;" +3= ";S
40  LET X=X+1
50  GO TO 20
```

Corre el programa. Te aparecerá una tabla como ésta:

```
0+3=3
1+3=4
2+3=5
3+3=6
4+3=7
```

★ Ahora vamos a hacer una tabla que ayude a ARTURO a multiplicar por 3. Cambia las líneas 20 y 30 del programa. Intenta hacerlo tú solo. Piensa que ahora vas a multiplicar en lugar de sumar. Si no te sale, no te preocupes. Estamos empezando.

GOTO

Mira la forma de hacerlo:

```
20  LET S=3 * X
30  PRINT " 3 *  ";X;" = ";S
```

Cambia estas líneas en el programa y al correrlo tendrás una pantalla como ésta:

```
3 * 0=0
3 * 1=3
3 * 2=6
3 * 3=9
3 * 4=12
3 * 5=15
```

★ La instrucción **GO TO** no sirve únicamente para volver a líneas de programa que ya habían sido ejecutadas, sino también para saltarnos líneas de programa e ir directamente a líneas posteriores a aquélla en la que está la instrucción **GO TO**.

O sea, que de la línea 40 podemos ir directamente —por ejemplo— a la línea 100.

Esto lo veremos mucho mejor más adelante. Por ahora, puedes entrenarte un poco con el programa "loco" de la página siguiente.

GOTO

Normalmente en un programa tú no harás una ordenación de líneas de programa tan absurda como ésta, pero te servirá para practicar y ver la velocidad con la que trabaja el ordenador.

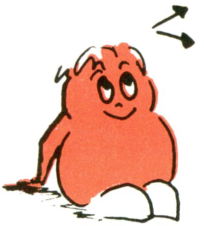
Escribe:

```
10  GO TO 40
20  PRINT "QUIEN ANDAS ";
30  GO TO 80
40  PRINT "DIME CON ";
50  GO TO 20
60  PRINT "QUIEN ERES"
70  END
80  PRINT " Y TE DIRE ";
90  GO TO 60
```

Fíjate en la línea 70. Sirve para que el ordenador se pare cuendo llegue allí. De lo contrario, el ordenador repetiría indefinidamente una serie de palabras (prueba a quitar la línea 70 y verás).

La instrucción END tiene, pues, su importancia. Algunos ordenadores utilizan STOP en lugar de END, pero su función es casi la misma.

¡Ah! En este tipo de programas tienes que poner mucha atención al colocar los espacios entre las comillas, para que te quede el texto bien escrito.



IF

La instrucción **GO TO** que acabamos de ver resulta todavía mucho más útil una vez que conozcas la instrucción **IF**. Esta instrucción permite que tu ordenador piense un poco por sí solo y tome sus decisiones.

Lógicamente, siempre lo hará sobre las órdenes o informaciones que tú le des.

★ Veamos cómo funciona esta instrucción en un programa:

```
10  PRINT "¿CUANTAS SON DOS MAS DOS?"
20  INPUT A
30  IF A=4 THEN GOTO 60
40  PRINT "TE HAS EQUIVOCADO"
50  GO TO 10
60  PRINT "HAS ACERTADO!!!"
```

IF

Este programa funciona así: En la línea 10 empieza la conversación contigo. En la línea 20 espera un dato que tú mismo le das a través del teclado.

Ahora viene la línea más interesante. En la línea 30, el ordenador se encuentra con una condición: Si el valor que le das a A es igual a 4, entonces saltará obligatoriamente a la línea 60, sin tener en cuenta las líneas 40 y 50. Y te dirá que has acertado.

Pero si A no es igual a 4, entonces el programa seguirá su curso normal. Hemos hecho que el ordenador tome una decisión siguiendo unas instrucciones que le hemos dado anteriormente.

Como ves, si A no es igual a 4, al llegar a la línea 40 te dice que te has equivocado y en la línea 50 vuelve a la línea 10 para preguntarte de nuevo.

Es decir, que al ordenador le pones unas condiciones. Se las pones con la instrucción **IF**. Si la condición se cumple, el ordenador hará una cosa (que tú le habrás indicado previamente). Si la condición no se cumple, hará otra (también indicada por tí).

En el ejemplo anterior, le poníamos la condición de que A fuera *igual* (=) a 4. Las condiciones que le quieras poner al ordenador pueden ser otras, tales como *menor que* (que se representa con el signo <), *mayor que* (que se representa con el signo >), *distinto de* (que se representa con el signo <>) y otras.

IF

★ El programa anterior podías haberlo escrito también de la siguiente manera:

```
10  PRINT " ¿CUANTAS SON DOS MAS DOS?"
20  INPUT A
30  IF A <> 4 THEN 60
40  PRINT "HAS ACERTADO!!!"
50  GO TO 80
60  PRINT "TE EQUIVOCASTE"
70  GO TO 10
80  END
```

Fíjate como ha sido necesario volver a organizar todas las líneas del programa al cambiar la condición. Pero el resultado es el mismo.

La condición que has puesto era: Si A es *distinto* de 4, vete a la línea 60.

Analiza bien cómo funciona cada uno de los programas.

IF

★ También puedes utilizar cadenas de caracteres (es decir, textos).

Vamos a escribir un programa casi igual que el primero, pero que nos pregunte como se llama ARTURO.

```
10  PRINT "¿COMO SE LLAMA NUESTRO AMIGO?"
20  INPUT A$
25  PRINT A$
30  IF A$="ARTURO" THEN 60
40  PRINT "TE HAS EQUIVOCADO"
50  GO TO 10
60  PRINT "HAS ACERTADO!!!"
```

Tienes que poner mucho cuidado al contestar al ordenador. Si no le contestas con los textos que él ya conoce, no entenderá tu respuesta. Vamos a poner a prueba al ordenador. Corre otra vez este programa y contéstale "ATURO", en lugar de "ARTURO". El ordenador desconoce la palabra ATURO, y la entenderá como un nombre distinto. Te dirá que te has equivocado, puesto que al no ser A$ exactamente igual a "ARTURO", el ordenador no salta a la línea 60 y sigue su lectura en la línea 40.

IF

★ Vamos a ver otro ejemplo, para que te acostumbres a usar esta instrucción.

El programa te parecerá un poco largo, pero es muy sencillo, porque ya conocemos todas las instrucciones que se usan en él.

Con este programa le resolvemos un problema a nuestro amigo ARTURO. El quiere comprar unos balones en una tienda. Solamente tiene 200 pesetas.

Hay dos tipos de balones. Rojos y azules. Los rojos valen 100 pesetas y los azules 90 pesetas. ARTURO tiene que decidirse entre comprar un balón rojo, uno azul o los dos. Y quiere saber cuánto dinero le va a sobrar en cada caso.

Hagamos el siguiente programa:

```
10  LET D=200
20  LET BR=100
30  LET BA=90
```

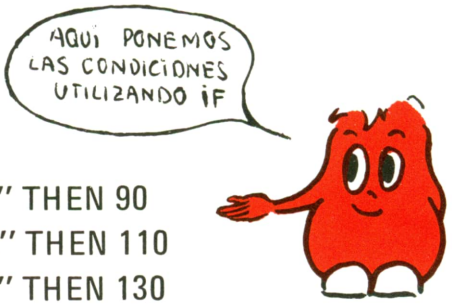
Con estas líneas llamamos D al dinero que tiene ARTURO y BR y BA a los precios de los balones rojos y azules, respectivamente.

IF

Sigue:

```
40  PRINT "DI CUAL COMPRAS: EL ROJO,
    EL AZUL O LOS DOS"
50  INPUT A$
55  PRINT A$
```

Con estas líneas le enseñamos al ordenador a preguntarnos y le indicamos que espere nuestra respuesta y la imprima. Sigue escribiendo:




```
60  IF A$ = "EL ROJO" THEN 90
70  IF A$ = "EL AZUL" THEN 110
80  IF A$ = "LOS DOS" THEN 130
85  PRINT "NO ENTIENDO. CONTESTA
    CORRECTAMENTE A MI PREGUNTA"
87  GO TO 40
```

Con estas líneas le indicamos al ordenador que actúe según sea nuestra respuesta. Como ves, según el balón elegido, al ordenador le mandas a una u otra línea del programa, para hacer todos los cálculos únicamente sobre ese balón y no otro.

IF

¿Qué pasa si tu respuesta no es igual a las previstas en las líneas 60, 70 y 80? Por ejemplo, si pones "verde" o "rojo" o incluso "azul" (en lugar de "el azul"), el ordenador no salta a ninguna línea y sigue leyendo el programa. Leerá las líneas 85 y 87 y te volverá a preguntar.

Sigue escribiendo:

```
 90  LET X = D - BR
100  PRINT "TIENES QUE PAGAR "; BR;
     " PESETAS Y TE SOBRAN "; X ; " PESETAS"
105  GO TO 150
```

El ordenador se dirige a la línea 90 solamente cuando tú has seleccionado el balón rojo. (Recuerda la línea 60.) Cuando llega a la línea 90, calcula X (que es el dinero que le sobra a ARTURO).

Para ello, el ordenador resta el precio del balón (BR) del dinero que llevaba (D). Acuérdate que estos valores los tenía el ordenador memorizados en las líneas 10 y 20.

IF

Seguimos:

```
110 LET X=D - BA
120 PRINT "TIENES QUE PAGAR "; BA;
    " PESETAS Y TE DEVOLVERAN ";
    X;" PESETAS"
125 GO TO 150
```

Como ves, el ordenador sólo llega a la línea 110 si eliges el balón azul (recuerda la línea 70). El ordenador hace todos los cálculos de igual forma que para el caso anterior, pero con los datos del precio del balón azul (BA).

No hará falta explicarte que, en el caso de que quieras los dos balones, el ordenador actuará de la misma forma, pero a partir de la línea 130.

¿Serás capaz de añadir tú las líneas 130 y siguientes?

Se pueden escribir así:

```
130 LET X=D - BR - BA
140 PRINT "TIENES QUE PAGAR "; BR +BA;
    " PESETAS Y TE DEVOLVERAN "; X;
    " PESETAS"
150 END
```

Corre el programa.

IF

★ Fijate en un detalle curioso de la línea 140. La cantidad total a pagar por los dos balones la hemos llamado BR + BA. El ordenador ha hecho el cálculo automáticamente.

Pero lo más correcto hubiera sido que este valor tuviera su correspondiente variable. Por ejemplo, podríamos haber dicho:

```
135 LET Z=BR+BA
```

Y en la línea 140 hubieramos escrito:

```
140 PRINT "TIENES QUE PAGAR ";Z;" PESETAS
    Y TE DEVOLVERAN ";X ;" PESETAS"
```

Hazlo de las dos formas y verás como el resultado es el mismo.

También habrás visto que en este programa utilizamos variables con dos letras para que nos sea más fácil identificarlas.

IF

NOTAS PARA EL ADULTO

- La instrucción **IF** permite establecer determinadas condiciones dentro del programa. Generalmente se utiliza para provocar un salto condicional dentro del mismo. Adopta entonces la forma **IF ... THEN** o **IF ... THEN GO TO** (según el tipo de ordenador). Es decir, el salto se produce únicamente si la condición puesta por **IF** se cumple. En caso contrario, se sigue ejecutando la línea siguiente a la de la instrucción **IF**.
- Esta instrucción es muy importante dentro de la programación, pero requiere una cierta madurez y capacidad de análisis por parte del niño.
  Si el niño lo encuentra confuso, limítese de momento a comentarle qué hace esta instrucción dentro de un programa, pero sin profundizar en el tema.
- Los ejemplos tratados en el texto son muy sencillos. Analícelos con el niño para hacerle ver que básicamente está utilizando instrucciones ya conocidas para él.
- Esta instrucción puede combinarse también con otras instrucciones en BASIC, como **PRINT, LET, STOP, END,** etc. Algunos ejemplos se comentarán más adelante (como el programa de la página 108). Otros ejemplos pueden ser:

```
IF X = 1 THEN PRINT "...."
IF A = B THEN STOP
IF M = 5 THEN LET C = C + 1
```

IF

SIGUE PROBANDO

★ Te vamos a explicar ahora un poco más sobre la instrucción **IF**. Tienes que prestar mucha atención, pero si no lo entiendes, no te preocupes. Ya sabrás hacerlo más adelante.

Verás. Como has comprobado, la instrucción **IF** permite poner unas condiciones dentro del programa. Si se cumplen, el programa salta a otra línea que nosotros queramos. Si no se cumplen, sigue su curso.

¿Te acuerdas del programa anterior? Con las líneas 60, 70 y 80 le poníamos condiciones al ordenador. Según se cumplirán esas condiciones, el ordenador saltaría a las líneas de programa 90, 110 ó 130.

En realidad, podríamos haber suprimido la línea de programa número 60. También tienes que quitar las líneas 85 y 87. Hazlo y verás que el resultado es el mismo.

Eso sucede porque si eliges el balón rojo no se cumple ninguna de las condiciones que habías puesto en las líneas 70 y 80 y el ordenador seguirá su curso normal y automáticamente leerá la línea de programa número 90. Sigue estas explicaciones que te damos con el programa a la vista y verás como lo entiendes.

Pero ¡Atención! Si no escoges "el azul" o "los dos", el ordenador también irá a la línea 90, pongas lo que pongas. Por ejemplo, pon "el verde" (que no existe) y verás.

IF

★ A ARTURO le han dado las notas de la primera evaluación. Si le dan notas entre 0 y 4 está suspendido. Si le dan notas entre 5 y 10 está aprobado. Ahora tú vas a ser el profesor de ARTURO en la asignatura de matemáticas. Dale la nota que quieras.

```
10  PRINT "¿QUE NOTA TIENE ARTURO?"
20  INPUT A
30  IF A<=4 THEN 60
40  PRINT "¡¡ARTURO HA APROBADO!!"
50  GO TO 70
60  PRINT " ... ARTURO HA SUSPENDIDO..."
70  END
```

El signo especial que aparece en la línea 30 significa "menor o igual a". Por eso la línea 30 es como si dijeramos: Si el valor que le demos a la variable A es menor o igual que 4, entonces vete a la línea 60.

Si el valor es distinto a esa condición, entonces el programa sigue su curso normal y se ejecutará la línea 40.

Corre el programa y dale diferentes notas a ARTURO.

IF

NOTAS PARA EL ADULTO

- Existen unos símbolos específicos por medio de los que establecemos una relación entre los términos de una expresión. Estos símbolos son similares a los utilizados en matemáticas. No obstante, existen pequeñas diferencias que deben ser respetadas escrupulosamente en lenguajes informáticos.
  Relacionamos a continuación los símbolos más usuales. Se llaman operadores lógicos y son:

  = es igual a
  < es menor que
  > es mayor que
  <= es menor o igual que
  >= es mayor o igual que
  <> es distinto de

- Invente algún programa sencillo en el que el niño pueda aplicar alguno de estos operadores lógicos.
- Analice cuidadosamente con el niño los comentarios de la página 75, especialmente en lo que se refiere a la supresión de la línea 60 del programa y sus consecuencias.
- Las instrucciones **STOP** y **END** tienen funciones similares, pero no idénticas. **STOP** permite la reanudación del programa mediante otra instrucción **CONTINUE**. **END** se utiliza para representar el final físico del programa (última línea). Sin embargo, fundamentalmente su inclusión está ligada a la posibilidad de terminar el programa en algún lugar distinto del final físico.

FOR / NEXT

Con todo lo que ya sabes, puedes hacer ya programas muy completos en lenguaje BASIC. Con un poco de imaginación y paciencia eres ya capaz de hacer cosas muy interesantes.

En este capítulo y en los siguientes vamos a explicarte algunas instrucciones que, aunque no son imprescindibles, son muy útiles para programar. Te facilitarán mucho las cosas y te ahorrarán mucho trabajo y tiempo.

Una de esas instrucciones se llama **FOR ... NEXT**. Verás la utilidad que tiene. Suponte que quieres hacer un programa que te de una lista de números del 0 hasta el 10. De acuerdo a lo que ya conocemos, tú harías el siguiente programa (intentalo tú primero):

```
10  LET A=0
20  PRINT A
30  LET A=A+1
40  IF A>10 THEN 60
50  GO TO 20
60  END
```

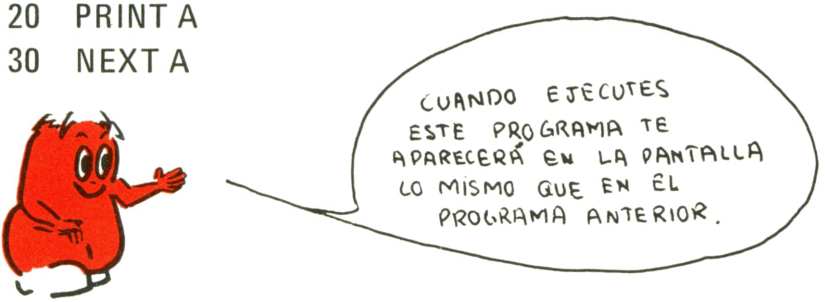
FOR / NEXT

En la línea 40 le hemos puesto al ordenador la condición de que cuando A tome un valor *mayor* que 10, salte a la línea 60 del programa, en donde se parará.

Mientras esta condición no se cumpla, el programa seguirá ejecutándose hasta la línea 50. Allí recibe la orden de volver a la línea 20, en donde irá imprimiendo el valor que tiene A en ese momento. Luego, en la línea 30, irá asignando nuevos valores a la variable A.

Pues bien, con la instrucción **FOR...NEXT** escribirás el mismo programa de la siguiente manera:

```
10  FOR A=0 TO 10
20  PRINT A
30  NEXT A
```



Córrelo y verás..Te has ahorrado muchas líneas de programa. Vamos a ver como funciona esta instrucción:

Las líneas 10 y 30 funcionan juntas. Es decir, que siempre que tengamos una línea FOR ... TO, tenemos que poner una línea con NEXT.

FOR / NEXT

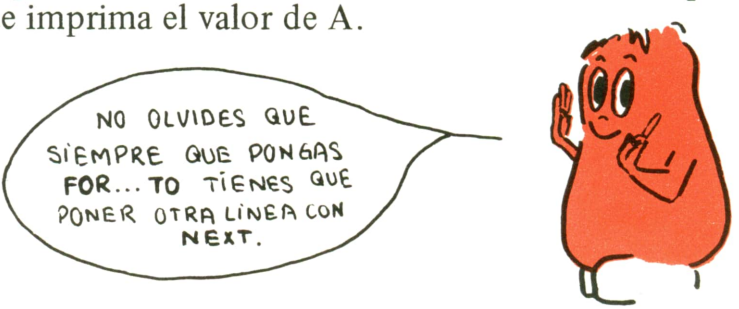
Fíjate en nuestro ejemplo. En la línea 10 le decimos al ordenador que a la variable A le vamos a dar valores entre 0 y 10. Es decir, que A va a valer primero 0, luego 1, luego 2, 3, 4, 5... hasta 10. Bien. El ordenador lo memoriza.

En la línea 20 le decimos al ordenador lo que queremos que haga con esos valores que va tomando A. En este caso, queremos que los imprima. En otros casos, podemos pedirle que haga cálculos o lo que queramos. Ya lo veremos en su momento.

En la línea 30, utilizando **NEXT**, le decimos al ordenador que tome los valores de A que tiene en su memoria. Es como si le dijéramos: "Toma uno detrás de otro todos los valores de A que te hemos dado con FOR ... TO".

Por eso esta instrucción tiene dos partes que van siempre juntas dentro del programa, aunque estén separadas por otras líneas de programa.

Entre la primera parte (FOR ... TO) y la segunda (NEXT), podemos indicarle al ordenador que haga lo que nosotros queramos. En este caso le hemos dicho simplemente que imprima el valor de A.



FOR / NEXT

Por tanto, el programa funciona así: En la línea 10 el ordenador memoriza que A va a valer 0, 1, 2, 3 ... hasta 10. En la línea 20, recibe la orden de imprimir dicho valor. En la línea 30 mandamos al ordenador tomar, uno detrás de otro, los valores de A.

O sea, que las líneas de este programa actúan así:

En la línea 10 . . . . . Se memorizan los valores que tendrá A
En la línea 20 . . . . . Imprime 0 (primer valor)
En la línea 30 . . . . . Toma el siguiente valor de A (que es 1) y vuelve a la línea 20

En la línea 20 . . . . . Imprime 1
En la línea 30 . . . . . Toma el siguiente valor de A (que es 2) y vuelve a la línea 20

En la línea 20 . . . . . Imprime 2

Y así sucesivamente hasta que A valga 10, momento en que en ordenador terminaría este proceso.

Este proceso se llama *bucle*. Dentro de un bucle se le puede pedir al ordenador que haga todos los cálculos que queramos. Los bucles son muy útiles cuando queremos repetir una operación que es siempre la misma porque, como has visto, el ordenador toma por sí solo los distintos valores con los que queramos trabajar.

FOR / NEXT

★ Veamos otra forma de utilizar esta instrucción. Haz el programa:

```
10  FOR A=0 TO 50
20  PRINT "HOLA";
30  NEXT A
```

Le estás diciendo al ordenador que imprima 50 veces la palabra HOLA.

La variable A no la estás utilizando para hacer cálculos con los valores que tome, sino simplemente como un "contador". Interesante ¿eh?

★ Fíjate en este programa de cálculo:

```
10  FOR A=1 TO 10
20  LET X=A * 5
30  PRINT X
40  NEXT A
```

Este programa quiere decir que el ordenador dará sucesivamente a la variable A valores desde 0 hasta 10, los multiplicará por 5 e imprimirá el resultado.

Ejecútalo.

FOR / NEXT

★ Completa el programa cambiando la línea 30 por:

```
30  PRINT A, X
```

Tendrás una pantalla como ésta:

1	5
2	10
3	15
4	20
5	25
6	30
7	35
8	40
9	45
10	50

En la primera columna aparecen los valores que va tomando A. En la segunda columna aparecen los sucesivos valores de X (resultado de multiplicar los sucesivos valores de A por 5).

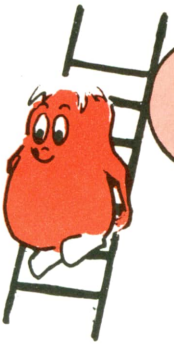
FOR / NEXT

NOTAS PARA EL ADULTO

- La instrucción **FOR/NEXT** ejecuta un bucle o secuencia iterativa que se inicia cuando la variable toma el valor inicial indicado en la instrucción y termina al alcanzar el valor final.
- Esta instrucción es muy utilizada en BASIC por su rapidez y comodidad, pero debe aprenderse dosificadamente. No fuerce al niño. Por el momento es más importante que sepa utilizarla en la práctica a la comprensión de todo el proceso desde un punto de vista conceptual.
- En los juegos que se ofrecen al final de este libro, aparecen dentro de los programas algunos bucles de este tipo. Su justificación es sencilla y conviene recomentar con el niño su repercusión en los programas.
- En un bucle **FOR/NEXT** es posible predeterminar el incremento del valor de la variable. Basta con añadir la instrucción **STEP**. Así, si quisiéramos que A incrementara su valor de dos en dos, escribiríamos:

```
10 FOR A=0 TO 10 STEP 2
```

- No olvide que una vez finalizado un bucle, el ordenador continúa el programa en la línea siguiente a la de la sentencia **NEXT.**
- Dentro de un bucle **FOR/NEXT** puede incluirse una instrucción (con **GOTO**) que obligue al programa a salirse del bucle, pero no se puede acceder al bucle desde el exterior del mismo.



GO SUB

Ya eres casi un programador experto. Te vamos a mostrar ahora una instrucción que no es imprescindible, pero que es también muy útil.

Se llama **GO SUB**, y va siempre acompañada de otra que se llama **RETURN**. ¿Te acuerdas de **GO TO**? Con **GO TO** obligábamos al ordenador a saltar a otra línea de programa. Con **GO SUB** pasa lo mismo, pero luego con **RETURN** hacemos que vuelva automáticamente otra vez al programa. Concretamente, a la línea siguiente a donde se encontraba la instrucción **GO SUB**.

¿Y esto para qué sirve?

Pues verás. Imagínate que tienes un programa y que hay un conjunto de instrucciones que se repite varias veces. Para no escribirlo todo todas esas veces, lo escribes solamente una (generalmente al final del programa) y mediante **GO SUB** haces que cuando sea necesario, el ordenador vaya a esa parte del programa.

GO SUB

Vamos a verlo con un ejemplo:

★ Suponte que estás rellenando un papel en donde te piden tu nombre y apellidos. El programa sería así:

```
 10  PRINT "NOMBRE"
 20  INPUT A$
 30  PRINT "PRIMER APELLIDO"
 40  INPUT B$
 50  PRINT "SEGUNDO APELLIDO"
 60  INPUT C$
 70  PRINT "FECHA DE NACIMIENTO"
 80  INPUT D$
 90  PRINT
100  PRINT
110  PRINT
120  PRINT A$ + B$ + C$
130  PRINT "NACI EL "; D$
```

Corre el programa. Cuando contestes, conviene que teclees un espacio en blanco después de dar tu nombre, apellidos, etc. Así queda más bonito cuando aparezca en pantalla y no se juntan las palabras.

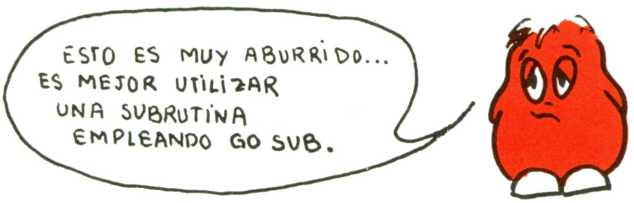
GO SUB

★ Supongamos que queremos que después de cada pregunta aparezca una línea de puntos para separar una pregunta de otra. Modificarías el programa añadiendo las líneas:

```
15  PRINT ".-.-.-.-.-.-.-.-.-"
35  PRINT ".-.-.-.-.-.-.-.-.-"
55  PRINT ".-.-.-.-.-.-.-.-.-"
75  PRINT ".-.-.-.-.-.-.-.-.-"
```

Hazlo y corre el programa. Introducir estas líneas ha sido muy aburrido, y tienes que prestar atención para no equivocarte poniendo puntos y rayas.

Si tuvieras un programa que te hiciera 20 ó 30 preguntas te pasarías el día entero escribiendo líneas como la línea 15.



Para evitarlo, utilizaremos una *subrutina.* Una subrutina es como un programa dentro del programa general. Casi siempre se ponen al final del programa general y puede tener muchas líneas.

GO SUB

★ En nuestro ejemplo, resolveríamos esta repetición de líneas con una subrutina como ésta:

```
200 PRINT ".-.-.-.-.-.-.-.-."
210 RETURN
```

Y ahora cambia las líneas 15, 35, 55 y 75 por líneas que digan :

```
GO SUB 200
```

Cada vez que el programa llega a una instrucción **GO SUB** salta a la línea 200, donde le mandamos imprimir la línea de puntos. En la línea 210 le obligamos a volver al programa principal. Vuelve siempre a la línea siguiente a aquella en que se encontraba **GO SUB**. En este caso, volverá sucesivamente a las líneas 20, 40, 60 y 80.

El ordenador sigue ejecutando el programa hasta que llega a otra instrucción **GO SUB**, y así sucesivamente.

★ Es necesario añadir una línea para indicarle al ordenador cuándo debe terminar el programa. Añade:

```
135 STOP
```

Tal vez te parezca que hay poca diferencia entre escribir cada línea de puntos en su sitio o indicarle al ordenador que se dirija a la subrutina. En nuestro ejemplo no hay

mucha diferencia, pero eso es porque hemos elegido una subrutina muy sencilla.

Imagínate que la subrutina fuera un complicado cálculo con muchas operaciones distintas y que tuvieras que escribirlas cada vez que, dentro del programa, fuera necesario efectuarlas.

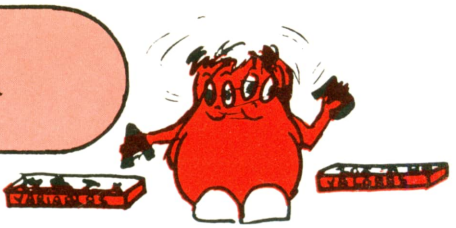
¡Seguro que prefieres utilizar **GO SUB**!

Además, no te olvides que una subrutina puede estar formada por muchas líneas de programa.

NOTAS PARA EL ADULTO

- Haga ver al niño las ventajas de la utilización de subrutinas. No obstante, no insista en este concepto hasta que el niño adquiera la suficiente soltura en el manejo y utilización de las demás instrucciones.
- Si lo cree oportuno, intenten conjuntamente el desarrollo de un ejemplo con subrutinas complejas, tanto con cálculos amplios como con textos muy extensos.
- Observe como las cadenas alfanuméricas pueden estar constituidas por grupos de palabras o números. La fecha de nacimiento es una cadena y por tanto hay que asignarla a una variable alfanumérica.

READ y DATA



Otra instrucción útil para ganar tiempo es la formada por **READ** y **DATA**.

Cuando en un programa hay que introducir muchos datos, las sentencias **LET** e **INPUT** nos hacen el trabajo muy pesado.

Fíjate en este programa:

```
10  LET A=5
20  LET B=7
30  LET C=4
40  LET D=2
50  LET E=12
60  LET S=A+B+C+D+E
70  PRINT S
```

Corre el programa y te aparecerá en pantalla la suma, que es 30.

READ y DATA

★ El mismo resultado lo obtenemos utilizando la instrucción **READ** y **DATA**.

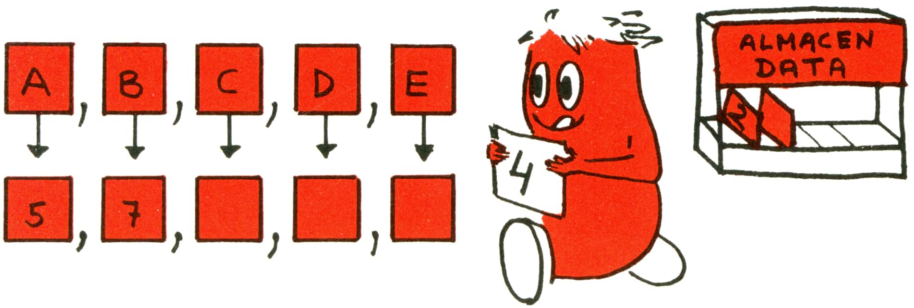
Con ellas, el programa se escribiría así:

```
10  READ A, B, C, D, E
20  DATA 5, 7, 4, 2, 12
30  LET S=A+B+C+D+E
40  PRINT S
```

El resultado es el mismo. Como ves, una sentencia **DATA** es como un almacén de datos, que están separados por comas. El ordenador no la tiene en cuenta a menos de que se encuentre una instrucción READ, que le obliga a leer los datos que están almacenados en la sentencia **DATA**.

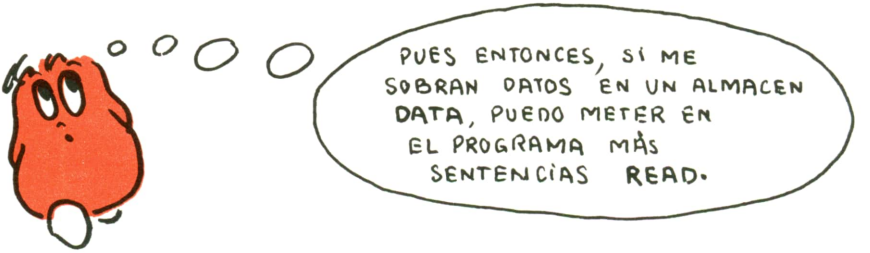
El ordenador toma los valores que estan en **DATA** y los asigna en el mismo orden a las variables que están en **READ**.

En nuestro ejemplo, A toma el valor 5, B toma el valor 7, etc.



READ y DATA

En un programa, una instrucción **DATA** puede tener más datos o valores que variables contenga una instrucción **READ**. Los datos que sobran quedan "a la espera" de que otras sentencias **READ** obliguen al ordenador a tomarlos.



Sobre nuestro ejemplo anterior, hagamos un programa con el que puedes ver como distintas instrucciones **READ** a lo largo del programa van tomando los valores que estaban en una instrucción **DATA**.

★ Fíjate también que esos valores pueden ser textos, números o expresiones. Vamos con el programa:

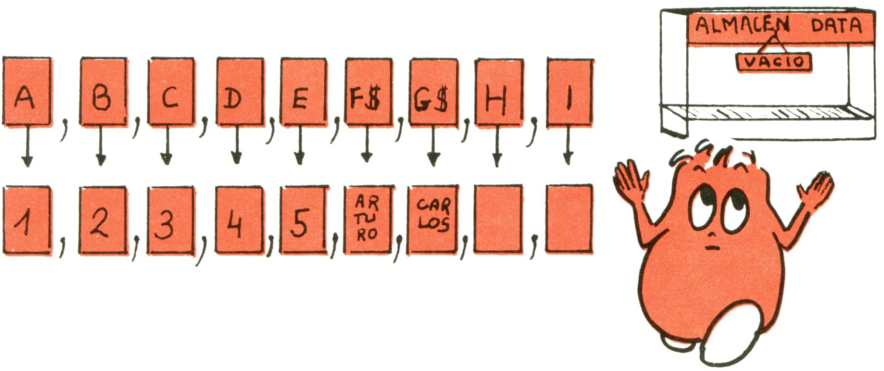
```
10  DATA 1, 2, 3, 4, 5, "ARTURO", "CARLOS"
20  READ A, B, C
30  READ D, E
40  LET S = A + B + C + D + E
50  READ F$, G$
60  PRINT F$; " y "; G$; " TIENEN "; S;
    " CARAMELOS"
```

READ y DATA

EN EL PROGRAMA ANTERIOR, LAS SENTENCIAS READ DE LAS LÍNEAS 20, 30 Y 50 VAN TOMANDO LOS VALORES QUE ESTÁN EN EL ALMACEN DATA DE LA LÍNEA 10.
FÍJATE COMO LAS VARIABLES DE LA LÍNEA 50 TIENEN EL SIGNO $ Y LES SERÁN ASIGNADAS LOS TEXTOS "ARTURO" Y "CARLOS".

Has visto que no hay problema si tenemos más datos que variables. Simplemente quedan a la espera de que el ordenador los lea a través de una nueva instrucción **READ**.

★ Pero si el ordenador se encuentra que tiene que leer un dato que no existe (es decir, que tenemos más variables que datos), aparecerá en la pantalla una indicación advirtiendote del error. El programa no sigue adelante. Compruébalo añadiendo a nuestro último programa la línea 55 READ H, I.

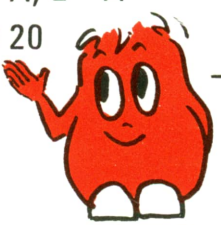


READ y DATA

Verás que el programa se interrumpe y te aparece un mensaje parecido a OUT OF DATA IN... indicándote seguidamente la línea en la que el ordenador encuentra variables a las que no puede asignar ningún valor.

★ Este mensaje puede aparecerte también cuando en un bucle, el ordenador se encuentre sin datos en una instrucción **DATA** y no pueda asignarlos a variables que esten en **READ**. Por ejemplo:

```
10  DATA 5, 7, 4, 2
20  READ A
30  PRINT A, 2 * A
40  GO TO 20
```



AQUÍ, UNA SOLA VARIABLE (A) VA A TOMAR LOS VALORES ALMACENADOS EN DATA.

En este programa el ordenador toma el primer valor (5) y lo asigna a la variable A. En la línea 30, imprime ese valor junto con dicho valor multiplicado por dos. En la línea 40 vuelve a comenzar el proceso asignándole sucesivamente a la variable A los valores almacenados en **DATA**. Cuando ya todos los valores están asignados, como la línea 40 remite al ordenador a la Instrucción **DATA**, aparecerá el mensaje OUT OF DATA.

READ y DATA

★ Si no se tiene cuidado, una instrucción de este tipo puede producir inconvenientes en un programa. Supongamos que se desean sumar los valores de una sentencia **DATA**. El programa podría ser:

```
10  DATA 5,7,4,2
20  LET S=0
30  READ A
40  LET S=S+A
50  GO TO 30
60  PRINT S
70  END
```

Este programa no funcionará nunca, pues está mal estructurado. Hubiera sido mejor cambiar el orden de las líneas 50 y 60. Tal y como está el programa, antes de llegar a la línea 60 e imprimir el resultado de la suma, el ordenador vuelve a la sentencia **DATA** y llega un momento en que se encuentra sin valores.

READ y DATA

★ Si el programa está muy complicado y resulta difícil cambiar el orden de las líneas, se puede utilizar un truco:

Se puede añadir un valor "falso" a la sentencia **DATA**. Es decir, un valor que a nosotros no nos interesa para nada en nuestro cálculo.

Además, ponemos una sentencia **IF** para que cuando el ordenador lea ese valor "falso" el programa salte a otra sentencia que nos convenga.

En nuestro ejemplo, transformaríamos la sentencia 10 así:

```
10  DATA 5, 7, 4, 2, 1000
```

y añadiríamos:

```
35  IF A = 1000 THEN 60
```

Así, cuando el ordenador lee el valor 1000, va directamente a la línea 60, donde imprime el valor de S acumulado hasta ese momento y termina el programa.

READ y DATA

NOTAS PARA EL ADULTO

- Esta instrucción asigna a las variables que figuran en **READ** los datos que aparecen en **DATA**. Insista ante el niño que una instrucción **DATA** puede abastecer a varias instrucciones **READ.**

- Conviene situar las sentencias **DATA** al principio o al final del programa.

- Esta sentencia sólo debe practicarse cuando ya estén perfectamente asimilados los conceptos vertidos en capítulos anteriores.

- Si el niño no ha adquirido la suficiente soltura, no insistan sobre esta sentencia. Siempre pueden volver sobre ella cuando lo consideren oportuno, ya que no es sino un valioso auxiliar en la programación.

REM

Las instrucciones que has conocido últimamente eran un poco complicadas. ¡Esta es muy fácil! Además, ayuda bastante a refrescarnos la memoria.

Una instrucción **REM** no afecta para nada el programa. El ordenador no la tiene en cuenta. Nos sirve a nosotros para no olvidarnos de lo que estamos haciendo.

Puede ocurrirte que cuando mires un programa que tú mismo has hecho, no te acuerdes muy bien de por qué hiciste algunas líneas de programa. Por eso puedes intercalar sentencias **REM** donde lo consideres oportuno, para que te refresquen la memoria cuando vuelvas a ver el programa dentro de algún tiempo. Es como una "chuleta".

REM

Por ejemplo, en el programa de la página 94 podríamos añadir las líneas:

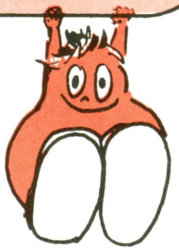
```
5    REM VALORES QUE TOMARA A
25   REM IMPRIME EL VALOR DE A Y
     EL DOBLE DE SU VALOR
```

Se pueden añadir todas las sentencias **REM** que se quiera. Ya sabes que no afectan al programa y sólo las puedes ver cuando listes el programa. Puedes poner el texto que tú quieras en una sentencia **REM.** Lo importante es que te recuerden, dónde tú lo precises, los pasos que has ido siguiendo al hacer un programa.

NOTAS PARA EL ADULTO

- Practique con el niño sobre programas anteriores añadiendo aclaraciones allí donde lo crean necesario.

INT



Esta instrucción o, mejor dicho, esta función, también será útil en programas de cálculo y en programas de juegos, como verás más adelante.

Lo que hace **INT** es quitar los decimales de un número, dejando sólo su parte entera.

De esta manera, el número 2,489 se quedaría en 2.

El número 5,246589 se quedaría en 5.

★ Vamos a poner un ejemplo. Escribe:

```
10  LET A=2.3489
20  PRINT A
```

Al correr el programa, aparece en pantalla 2.3489. Fíjate en algo muy importante: En lugar de utilizar la coma para separar la parte entera de la decimal, utilizas un punto. Esto debes hacerlo siempre en Informática.

★ Añade ahora la línea:

```
15  LET A=INT (A)
```

INT

Corre el programa.

En pantalla te aparece el número 2. Has dejado únicamente la parte entera.

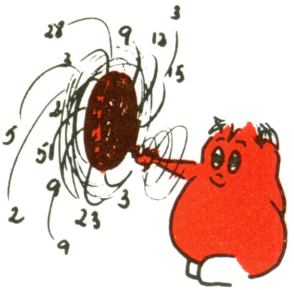
Fíjate que la función **INT** no redondea. Simplemente, deja la parte entera, o sea, que con **INT** el número 2.9 se convierte en el número 2 y no en el número 3.

Como te hemos dicho antes, **INT** es de mucha utilidad en lenguaje BASIC. Ya lo irás viendo con el tiempo. En el próximo capítulo verás algunas cosas interesantes sobre su utilización.

NOTAS PARA EL ADULTO

- Sobre el programa anterior, haga que el niño modifique la línea 10 varias veces, con números de su elección.
- Es probable que el niño no se encuentre especialmente motivado por esta función. Anticípele que tiene gran importancia en los juegos y en cálculos y programas más complejos.
- Remarque la importancia de utilizar un punto para separar la parte entera de la decimal. Además, tenga en cuenta que en Informática no se utiliza el punto para diferenciar unidades de mil. Así, dos mil cien se escribiría 2100.

RND



¿Sabes lo qué es el azar, o la casualidad?

Muchas veces tú dices: "Esto ha pasado por casualidad". Es decir, quieres decir que también podía *no haber pasado*. O podía *haber pasado otra cosa.*

Pues bien, con la función **RND** puedes introducir el azar en un programa. Y eso ¿para qué sirve? Pues para mucho. Por ejemplo, para jugar al escondite con el ordenador, como veremos más tarde. También, para simular con el ordenador sucesos que ocurren en la vida real y en los que interviene el azar.

★ Pero vayamos poco a poco. Con la instrucción **RND** se hace que el ordenador fabrique o genere números al azar (aleatorios) entre 0 y 1 (sin incluir el 1). Por ejemplo, escribe:

```
10  LET A = RND (X)
20  PRINT A
```

RND

Cada vez que corras el programa, en la pantalla te aparecerá un número igual o mayor que 0 y menor que 1. Por ejemplo:

0.53118986
0.83547393
0.97816467
0.24576398

★ Para que el ordenador nos de números al azar comprendidos entre 0 y 10 (por ejemplo), tendremos que multiplicar por 10 el valor de RND (X). Cambia la línea 10 por:

```
10  LET A = RND (X) * 10
```

RND

Añade:

```
30  GO TO 10
```

Tendrás una lista de números al azar entre 0 y 10 (sin incluir el 10). Por ejemplo:

```
0.18976534
4.00789654
1.36962891
3.00000000
8.34698762
```

SI QUIERES TENER NÚMEROS ENTRE 0 Y 20, MULTIPLICA RND(x) POR 20. ENTRE 0 Y 30, MULTIPLICA RND(x) POR 30. Y ASÍ SUCESIVAMENTE.

¿Te acuerdas de la instrucción **INT**? Te va a ser muy útil ahora.

Con ella puedes hacer que sólo quede la parte entera de estos números.

★ Cambia la línea 10 por:

```
10  LET A=INT (RND (X) * 10)
```

CON INT QUITAMOS LAS CIFRAS DESPUÉS DEL PUNTO Y DEJAMOS SÓLO LA PARTE ENTERA.

Tendrás una lista de números enteros entre 0 y 10, excluyendo el 10.

RND

★ Suponte ahora que quieres analizar los números que saldrían al lanzar un dado. Siguiendo las fórmulas del ejemplo anterior, escribirías:

```
10  LET A=INT (RND (X) * 6)
```

Pero ¡ ¡atención!! Esta sentencia sólo te da valores enteros entre 0 y 6, excluido el 6. Y además, el dado no tiene ceros. Por eso, tendrás que añadir una unidad, para que tengas efectivamente números del 1 al 6, ambos inclusive. Escribe pues:

```
10  LET A=INT (RND (X) * 6) +1
20  PRINT A ;"  ";
30  GO TO 10
```

Al correr el programa, tendrás en la pantalla una lista con los números que podían haber salido al tirar un dado.

Has reproducido con el ordenador lo mismo que te puede salir en la vida real cuando juegas a los dados. Quita la línea 30 y cada vez que corras el programa, es como si tiraras un dado.

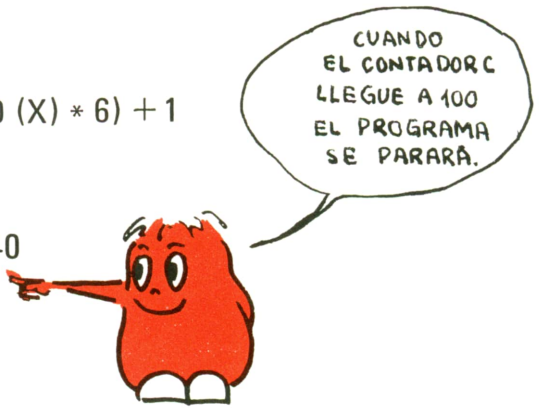
RND

★ Suponte que queremos hacer exactamente 100 tiradas de dados para ver qué números nos pueden salir en esas 100 tiradas. Para hacerlo, modificaremos el programa añadiendo un "contador". Cuando el contador llegue a 100, el programa se parará. Añade las líneas:

```
5   LET C=0
25  LET C=C+1
26  IF C=100 THEN 40
30  GOTO 10
40  END
```

Tendrás ahora un programa como éste:

```
5   LET C=0
10  LET A=INT (RND (X) * 6) +1
20  PRINT A;"  ";
25  LET C=C+1
26  IF C=100 THEN 40
30  GO TO 10
40  END
```



RND

El programa funcionaría así: En la línea 5 pones el contador a cero. En las líneas 10, 20 y 30 el ordenador va dando al azar e imprimiendo números comprendidos entre 1 y 6. En la línea 25 hacemos que el contador vaya aumentando de uno en uno. En la línea 26 indicamos al ordenador que cuando el contador llegue a 100, salte a la línea 40 con lo que el programa se para.

Analiza bien estas aclaraciones con el programa a la vista. Luego, ejecútalo.

★ ¡Y ahora viene lo bueno! Tú sabes que hay las mismas posibilidades de que al tirar un dado te salga un número u otro.

En teoría, podríamos pensar que de 100 tiradas, un número determinado (por ejemplo, el 5) se repetiría 16 veces. (Claro. 100 tiradas dividido entre 6 números que pueden salir = 16 posibles repeticiones por número).

Pero en la práctica, el azar hace que pueda salir más o menos veces. Podemos hacer pruebas tirando 100 veces un dado, para ver qué pasa. Pero eso es muy aburrido y lento.

Con el ordenador podemos simular la tirada de los dados. De forma mucho más rápida, podemos ver que pasaría tirando 100 veces un dado y contando cuántas veces sale un número (por ejemplo, el 5).

RND

★ Para hacerlo, sólo tenemos que añadir al programa anterior unas líneas que nos cuenten, dentro del programa, cada vez que aparece el número 5.

Añade:

```
7    LET P=0
22   IF A=5 THEN LET P=P+1
35   PRINT C, P
```

y cambia la línea 26 por:

```
26 IF C=100 THEN 31
```

Añade también:

```
31   PRINT
32   PRINT
33   PRINT
34   PRINT
```

Con ello tendrás una pantalla final más clara, pues el resultado final queda separado de los otros números.

Con la línea 7 inicias el contador que marca las veces que aparece el número 5. Con la línea 22, le dices al ordenador que cada vez que el número sea 5, añada una unidad al contador.

RND

Con la línea 35 haces que, al final, el ordenador te imprima el número de tiradas y las veces que ha salido el número 5.

Como has visto, tienes que cambiar un poco la línea 26 para que el programa esté bien estructurado.

El programa final te habrá quedado así:

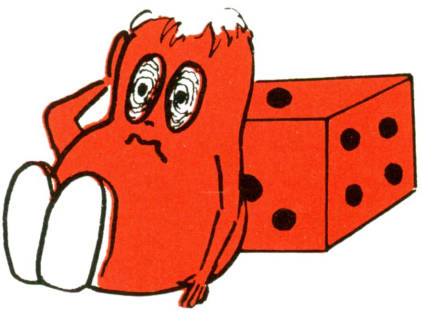
```
5   LET C=0
7   LET P=0
10  LET A=INT (RND (X) * 6) +1
20  PRINT A;"  ";
22  IF A=5 THEN LET P=P +1
25  LET C=C +1
26  IF C=100 THEN 31
30  GO TO 10
31  PRINT
32  PRINT
33  PRINT
34  PRINT
35  PRINT C, P
40  END
```

Ejecútalo y verás. Al final, el ordenador te dirá que ha probado 100 veces y el número de veces que salió el número 5.

RND

★ Fíjate la ventaja que tienes cuando trabajas con un ordenador. Suponte que quieres ver cuantas veces sale el número 5 en 1000 tiradas.

ARTURO intentó tirar 1000 veces el dado pero se cansó cuando llevaba 168. Con el ordenador es todo mucho más rápido.



Quita la línea 20 (para que no te escriba todos los números que van saliendo) y cambia la línea 26 por:

```
26  IF C=1000 THEN 35
```

Cambia también la línea 35 poniendo:

```
35  PRINT "EN ";C;" TIRADAS HAN SALIDO ";
    P;" CINCOS"
```

Corre el programa. Tendrás que esperar un poco, pero mucho menos que si esperaras a que ARTURO te diera el resultado.

RND

★ ¿Serías capaz de modificar el programa para que el ordenador te diga además cuántos cuatros o seises han salido en las 1000 tiradas? Es muy fácil. Sólo tienes que tener en cuenta que tendrás que poner más contadores para estos números. Por ejemplo:

```
6   LET Q=0
23  IF A=6 THEN LET Q=Q+1
```

y cambia la línea 35 por:

```
35 PRINT "EN "; C; " TIRADAS HAN SALIDO ";
   P; " CINCOS y "; Q; " SEISES"
```

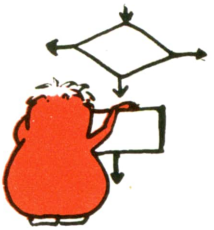
Haz tú las modificaciones para los demás números.

RND

NOTAS PARA EL ADULTO

- Al igual que en capítulos anteriores, esta instrucción debe tratarse secuencial y paulatinamente. Lo único importante es que el niño capte el concepto de aleatoriedad que se obtiene a través de **RND.**
- Motive al niño después de cada paso animándole a descubrir las ventajas que le ofrece el paso siguiente. Plantee todo como un experimento.
- La función **RND** no se expresa igual en todos los ordenadores. Generalmente, en BASIC se exige que después de **RND** se sitúe un valor entre paréntesis. Este valor puede ser, sencillamente X. No se preocupe si ya existe esta variable en el mismo programa en que utilice la instrucción **RND.**
- Consulte previamente el manual de instrucciones de su ordenador personal. Estudie con atención la forma de expresar la función **RND** y haga las modificaciones oportunas en los programas expuestos en este capítulo.
- Algunos ordenadores requieren que se incluya al principio del programa el comando RANDOMIZE, a fin de obtener diferentes series de números aleatorios cada vez que se ejecute el programa. De lo contrario, siempre darán la misma serie. Así, se escribiría:

```
10  RANDOMIZE
20  LET A=RND (X)
30  PRINT A
```



Diagramas

Conviene que sepas que, a la hora de programar, es útil utilizar unos dibujos o diagramas para tener las ideas muy claras y organizar bien el trabajo que vamos a hacer.

Además, estos diagramas te servirán para comprender, con un poco de atención, como está construido o estructurado el programa. Puedes ver enseguida cómo va a funcionar cuando lo estés ejecutando.

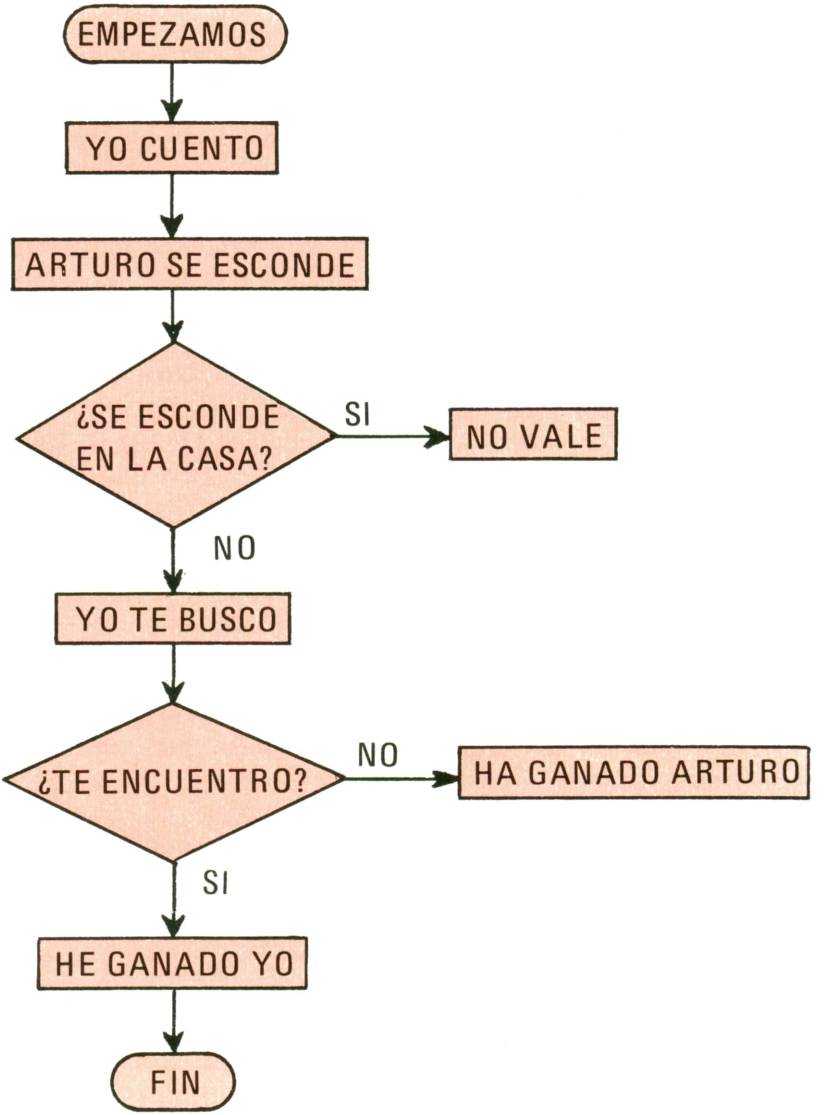
Lo mejor es hacer un diagrama antes de hacer el programa, para tener muy claro como vamos a explicarle al ordenador lo que tiene que hacer. Sobre todo cuando el programa es un poco complicado.

No vamos a estudiarlos en profundidad. Solamente vamos a ver unos ejemplos muy fáciles para que te acostumbres a conocerlos.

¿Te acuerdas del juego del escondite y de las reglas del juego que marcábamos al principio de este libro?

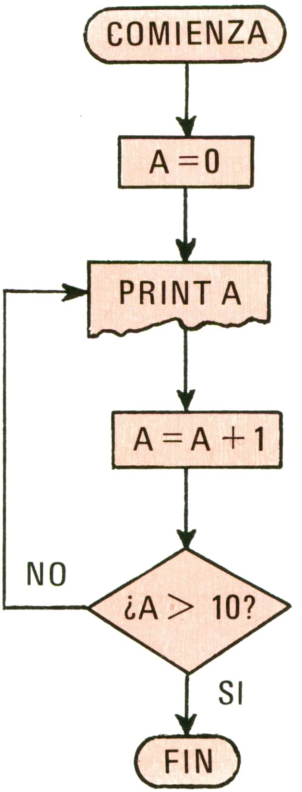
Si tuviéramos que hacer un diagrama de aquél juego, haríamos algo parecido a lo que te mostramos en la página siguiente. Míralo con atención y verás como todo el desarrollo es muy lógico.

Diagramas


EMPEZAMOS
YO CUENTO
ARTURO SE ESCONDE
¿SE ESCONDE EN LA CASA?
SI
NO VALE
NO
YO TE BUSCO
¿TE ENCUENTRO?
NO
HA GANADO ARTURO
SI
HE GANADO YO
FIN

Diagramas

★ Tomemos un programa muy sencillo que nos escribe los números naturales del 0 al 10. El diagrama sería:



Diagramas

El programa escrito quedaría así:

```
10  LET A=0
20  PRINT A
30  LET A=A+1
40  IF A> 10 THEN 60
50  GO TO 20
60  END
```

Corre el programa y verás como aparece una lista de números del 0 al 10. Te habrá parecido muy fácil pues ya conoces muy bien todas las instrucciones.

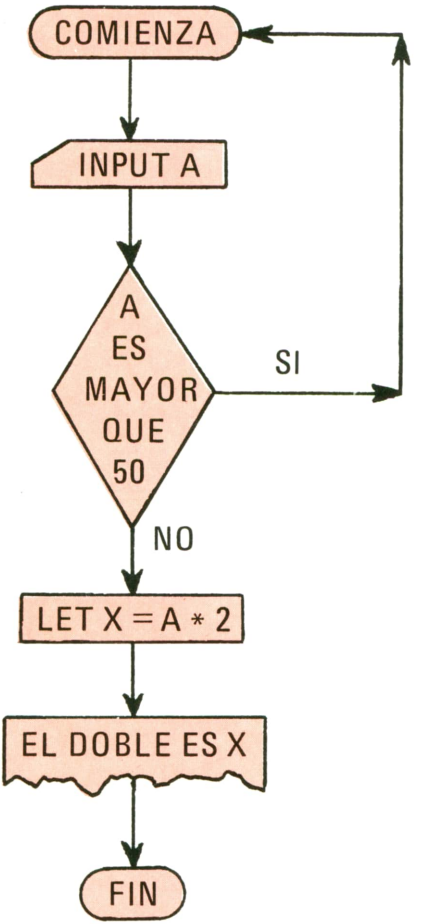
★ ¿Cómo sería el diagrama de flujo de un programa que calculara el doble de un número comprendido entre 0 y 50?

El programa sería:

```
10  PRINT "DI UN NUMERO ENTRE 0 y 50"
20  INPUT A
30  IF A>50 THEN 10
40  LET X=A*2
50  PRINT" EL DOBLE ES ";X
60  END
```

Y el diagrama sería:

Diagramas


COMIENZA
INPUT A
A ES MAYOR QUE 50
SI
NO
LET X = A * 2
EL DOBLE ES X
FIN

Diagramas

NOTAS PARA EL ADULTO

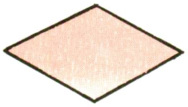
- Los símbolos convencionales que se usan en los diagramas de flujo pueden variar ligeramente, pero los más comúnmente admitidos son:

PRINCIPIO O FIN DE PROGRAMA

ENTRADA O SALIDA DE DATOS (INPUT)

ASIGNACION DE VARIABLES O REALIZACION DE CALCULOS (LET)

IMPRESION DE RESULTADOS (PRINT)



DECISIONES LOGICAS (IF ...)

- Aunque es conveniente iniciar al niño en el desarrollo de diagramas de flujo, su explicación detallada escapa a los objetivos de este libro.

- Cuando el niño domine las instrucciones en BASIC que hemos estudiado en este libro y su repercusión dentro de un programa intenten desarrollar diagramas sobre algunos de los programas ya vistos.

Juegos

En este capítulo vamos a ver unos pocos juegos o aplicaciones que pueden resultarte divertidos, útiles e interesantes.

Ojalá te sirvan para desarrollar luego tus propios juegos.

En todos los programas se utilizan instrucciones y sentencias que ya conoces bien. Intenta analizarlos primero para que veas como funcionan.

APRENDIENDO A MULTIPLICAR

Este programa te ayudará de manera divertida a practicar la tabla de multiplicar.

```
10  PRINT "DI UN NUMERO ";
20  INPUT N
30  PRINT N
```

Juegos

```
 40  FOR I=1 TO 10
 50  PRINT N; " * "; I; " =";
 60  INPUT M
 70  LET P=N * I
 80  IF M=P THEN 110
 90  PRINT "¡FALLASTE. INTENTALO DE NUEVO!"
100  GO TO 50
110  PRINT "¡MUY BIEN!"
120  NEXT I
130  END
```

NOTAS PARA EL ADULTO

- En este programa se practican las instrucciones **PRINT, INPUT, FOR NEXT, IF** y **GOTO.**

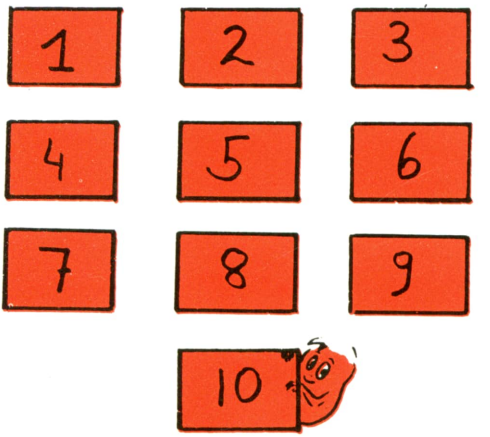
Juegos

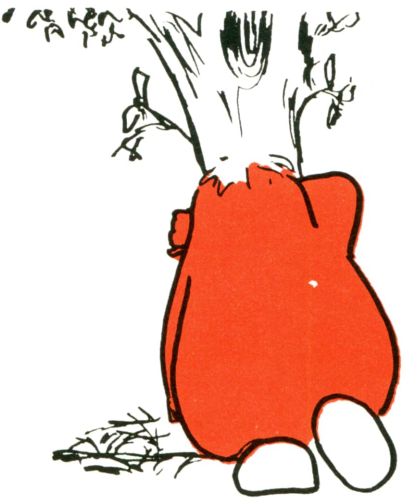
JUGANDO AL ESCONDITE CON ARTURO Y EL ORDENADOR

Como te prometimos al principio del libro, vamos a jugar al escondite con ARTURO. El programa es muy sencillo. Si lo analizas un poco, lo comprenderás perfectamente y verás que hubieras podido hacerlo tú mismo.

El juego consiste en lo siguiente: ARTURO se va a esconder en cualquiera de los sitios que hay en el jardín. Sólo puede esconderse en 10 sitios, numerados del 1 al 10. Si se esconde en la casa el juego se acaba en cuanto tú empieces a buscarle.

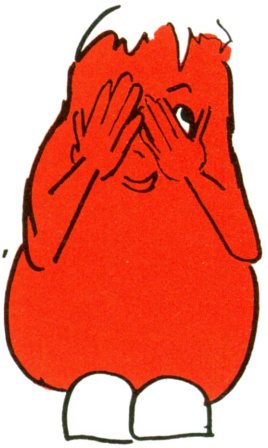
Cuando lo encuentres, el ordenador te dirá cuántos intentos has hecho para encontrarle. Cada vez que juegues se esconderá en un sitio distinto. ¡A ver quien lo encuentra primero!





Juegos

```
 5  REM ESCONDITE
10  LET A=0
15  RANDOMIZE
20  LET R=INT (RND (X) * 10) +1
25  PRINT "ESCRIBE EL NUMERO DE ESCONDITE"
30  PRINT "¿DONDE ESTA ARTURO?";
35  LET A=A +1
40  INPUT N
50  PRINT N
60  IF R=3 THEN 200
70  IF R=N THEN 100
80  PRINT "AQUI NO ESTA ... "
90  GO TO 30
100 PRINT " ¡LO HAS ENCONTRADO!"
101 PRINT
102 PRINT
103 PRINT "EN  "; A; "  INTENTOS"
105 GO TO 220
200 PRINT "NO VALE. ARTURO SE HA ESCONDIDO
    EN LA CASA"
210 PRINT "EMPIEZA OTRA VEZ"
220 END
```

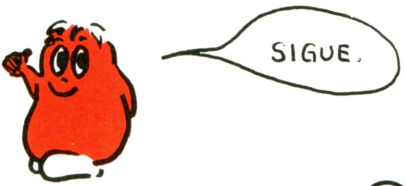


Juegos

ADIVINANDO NUMEROS

Esto es un programa parecido al escondite. El ordenador pensará un número entre 1 y 100. Tienes que adivinarlo. Al responderte, te dará pistas. ¡Descubre el número en pocos intentos!

```
   5   REM ADIVINANDO
  10   LET A=0
  20   LET R=INT (RND (X) * 100) +1
  30   PRINT "PIENSO UN NUMERO ENTRE 1 y 100"
  40   PRINT "INTENTA ADIVINARLO"
  50   PRINT "DI UN NUMERO"
  60   INPUT X
  63   PRINT
  65   PRINT X
  70   LET A=A +1
  80   IF X<R THEN 2000
  90   IF X=R THEN 3000
1000   PRINT
1005   PRINT
1010   PRINT "FALLASTE. MI NUMERO ES MENOR"
1020   GO TO 50
2000   PRINT
2010   PRINT
```



```
2020  PRINT "FALLASTE. MI NUMERO ES MAYOR"
2030  GO TO 50
3000  PRINT
3010  PRINT
3020  PRINT "¡¡ACERTASTE!!"
3030  PRINT
3040  PRINT "MI NUMERO ES EL  "; R
3050  PRINT
3060  PRINT "LO HAS AVERIGUADO EN  "; A;
      "  INTENTOS"
3070  PRINT
3080  IF A >= 5 THEN 3100
3090  PRINT "BUEN RESULTADO"
3093  PRINT
3095  GO TO 3105
3100  PRINT "NO ESTA MAL, PERO INTENTA
      MEJORARLO"
3105  PRINT
3110  END
```

Juegos

EL ORDENADOR SABIO

El ordenador puede adivinar tus pensamientos. Compruébalo con este programa. Toma papel y lápiz y haz lo que te diga.

```
 10 PRINT "VOY A ADIVINAR TU NUMERO"
 20 PRINT
 30 PRINT "PIENSA UN NUMERO"
 40 PRINT " PERO NO ME LO DIGAS"
 50 GO SUB 300
 60 PRINT "MULTIPLICALO POR 6"
 70 GO SUB 300
 80 PRINT "SUMA 10 AL RESULTADO"
 90 GO SUB 300
100 PRINT "DIVIDE EL RESULTADO ENTRE 2"
110 GO SUB 300
120 PRINT "RESTALE 5 AL RESULTADO"
130 GO SUB 300
140 PRINT "DIME EL RESULTADO FINAL"
150 INPUT X
160 LET Z = X + 5
170 LET Y = Z * 2
180 LET H = Y - 10
```

```
190 LET N=H/6
200 PRINT
210 PRINT
215 PRINT
216 PRINT
220 PRINT "PENSASTE EL NUMERO ";N
225 PRINT
230 STOP
300 PRINT "SI ESTAS LISTO"
310 PRINT "PULSA ENTER"
320 INPUT A$
330 PRINT
340 PRINT
350 RETURN
```

NOTAS PARA EL ADULTO

- En este programa se hace uso de la instrucción **GO SUB**, además de las anteriores.

BASIC avanzado

Ll
ate

es, el
uda, y
decuada-

l forma que
res que carez-

n profundizar más
deden consultar la in-
, publicada en esta mis-

Seguimos

Aquí tienes el resumen de lo que ya sabemos, para que te sirva de recordatorio:

PRINT Manda imprimir o escribir en la pantalla todo lo que le indiques.

LET Te permite dar a una variable el valor que tú quieras.

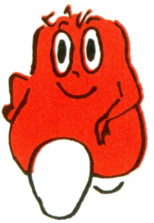
INPUT Hace que el ordenador espere un dato que tú le puedes dar a través de tu teclado mientras se está ejecutando o corriendo el programa.

GOTO Con ella le das una orden tajante al ordenador para que se dirija obligatoriamente a la línea de programa que tú le indiques.

IF ... THEN Permite que le des unas instrucciones al ordenador para que, dependiendo de si se cumplen o no unas condiciones que tú mismo le marcas, haga una cosa u otra.

Seguimos

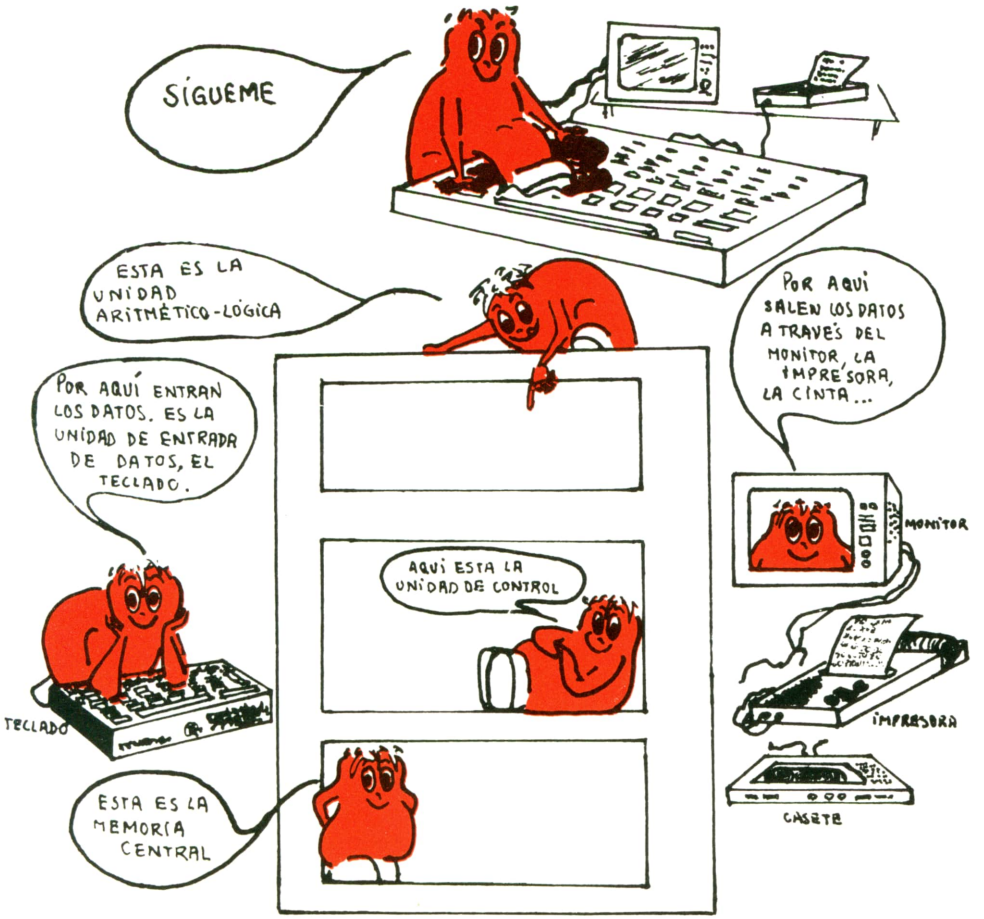
FOR/NEXT	Con esta instrucción preparamos al ordenador para efectuar un bucle y repetir una misma acción un número determinado de veces.
GOSUB	Manda al ordenador a una determinada subrutina. Con la instrucción RETURN el ordenador vuelve al programa principal.
READ/DATA	Con la instrucción DATA almacenamos en el programa unos datos que serán leídos por una instrucción READ.
REM	Te permite introducir comentarios en un programa, sin alterarlo.
INT	Quita los decimales de un número, dejando sólo la parte entera.
RND	Con ella podemos introducir el azar en un programa.
DIAGRAMAS	Nos permiten ver con un esquema cómo va a estar construido un programa.



Mirando dentro

Ahora que ya sabes hacer tus propios programas vamos a analizar cómo y por qué interpreta el ordenador tus instrucciones. Es decir, como trabaja un ordenador por dentro.

Ya sabes que un ordenador memoriza todas las líneas de programa que tú le das. Entremos en el ordenador para ver dónde las guarda y qué hace con los datos que tú introduces.



Mirando dentro

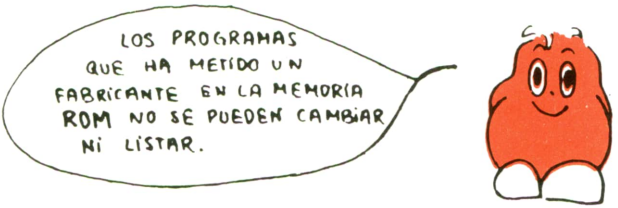
ARTURO tardó algún tiempo en aprenderse estos nombres que ves en el dibujo, pero lo importante no es saberse los nombres sino entender cómo funciona cada una de estas partes del ordenador.

Cuando tú, a través del teclado, metes unos datos en el ordenador, éstos van directamente a la MEMORIA CENTRAL. ¿Recuerdas que al ordenador hay que decírselo casi todo y que hay que enseñarle primero lo que debe hacer para que al final el programa dé el resultado que tú querías?

Bien. Hay que decírselo casi todo, pero algunas cosas ya las sabía. Fijate bien en esto. Tú le dabas las instrucciones con unas líneas numeradas para que él las leyera por orden. Incluso podías darle las instrucciones con la numeración desordenada y él las ordenaba automáticamente. También has hecho programas para que el ordenador sume, multiplique... Pero tú no le enseñaste ni a sumar ni a poner las líneas en orden.

Dentro de la memoria central está la memoria ROM. En ella el ordenador tiene memorizados todos los datos y programas que necesita para saber contar, ordenar.... Todos esos datos los introdujo el fabricante en la memoria ROM

para que siempre estén ahí guardados. Cuando tú haces un programa en el que hay una suma, estás diciendo al ordenador "Mira en tu memoria ROM cómo hacer esta operación, y hazla".



Pero dentro de la memoria CENTRAL también existe la memoria RAM. Ella es la que se encarga de memorizar todas las instrucciones y datos que tú metes en el ordenador a través del dispositivo de entrada de datos, a través del teclado. Por eso tú puedes introducir datos y leerlos después. Porque se guardan en la memoria RAM, a la que tú puedes llegar y modificar. Cuando listas un programa, estás preguntando al ordenador: ¿qué tienes en la memoria RAM?

¡Ah! Y cuando desenchufas el ordenador, se pierde todo lo que esté almacenado en la memoria RAM, pero se sigue guardando el contenido de la memoria ROM, pues de lo contrario el ordenador no sabría como trabajar para tí cuando le pidas que haga operaciones o que interprete tus instrucciones.

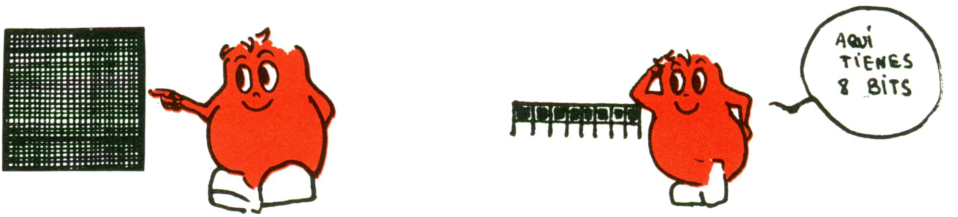
Mirando dentro

Cada modelo de ordenador tiene una capacidad de memoria diferente. Algunos ordenadores son capaces de guardar en su memoria central programas larguiiiiisimos o muchos programas y otros ordenadores no tanto.

Para saber si un ordenador tiene poca o mucha memoria, nosotros hablamos de K. Seguro que has oído decir "este ordenador tiene 32 K de memoria".

¿Qué es una K? Verás:

La memoria central por dentro podríamos verla como unas celdillas. Cada celdilla es un BIT



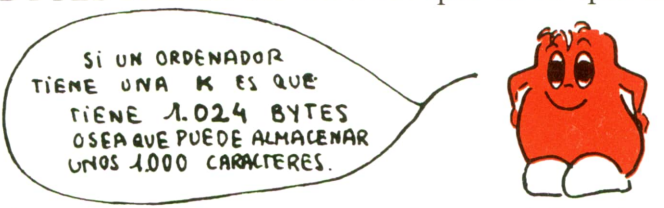
A cada conjunto de 8 bits le llamamos BYTE

En cada BYTE se almacena un carácter. Es decir, que el ordenador utiliza un BYTE, o sea, 8 bits, para cada carácter.

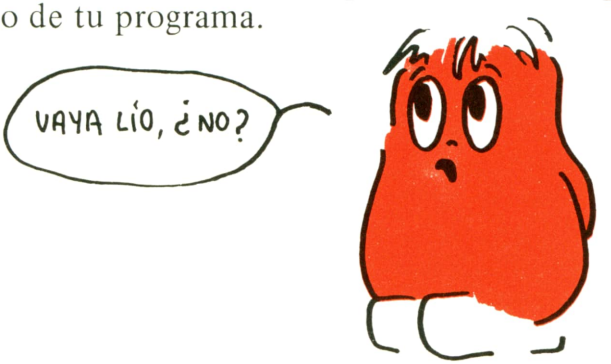
Mirando dentro

Así, el ordenador utilizaría 8 celdillas (8 bits) para la letra A, o la B, o para el número 2 ó para el número 7, etc.

1024 BYTES forman un KBYTE que es lo que llamamos K.



★ Pero sigamos viendo qué pasa dentro del ordenador. Los datos e instrucciones van primero a la UNIDAD CENTRAL DE MEMORIA. De ahí pasan a la UNIDAD DE CONTROL. La UNIDAD DE CONTROL se encarga de interpretar las instrucciones y órdenes que le has dado en el programa. La UNIDAD ARITMETICO LOGICA es la que hace los cálculos, las comparaciones, etc. El resultado de estos cálculos o comparaciones pasa otra vez por la UNIDAD DE CONTROL y va a parar a la MEMORIA CENTRAL, de donde va por la UNIDAD DE SALIDA al exterior. Es entonces cuando tú puedes ver en la pantalla el resultado de tu programa.



Mirando dentro

No te preocupes. Con un ejemplo lo vamos a ver en seguida. Fíjate en el programa:

```
10  LET A=5
20  LET B=3
30  LET C=A+B
40  PRINT C
50  STOP
```

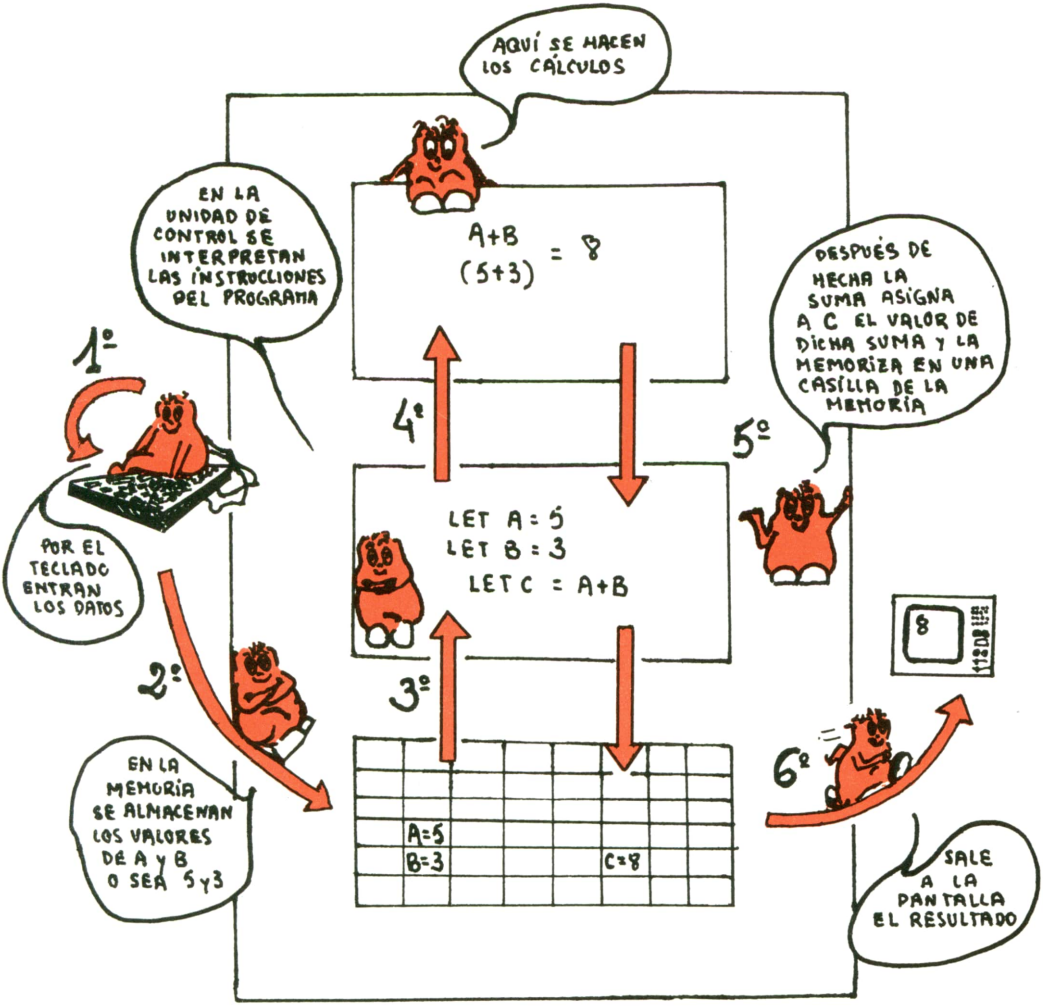
Y ahora vas a ver en el dibujo de la página siguiente, cómo este programa es interpretado y ejecutado en las distintas partes del ordenador.

No te preocupes si no entiendes muy bien todo esto. En realidad, lo importante es que tengas una idea de cómo funciona un ordenador y que cuando oigas alguna de estas palabras sepas aproximadamente de qué se está hablando.

Todo lo que puedes ver por dentro y por fuera del ordenador (cables, circuitos, teclado, pantalla, impresora, etc.) es lo que se llama HARDWARE (o equipo físico).

Los programas, las ideas... es lo que se llama SOFTWARE (o logical).

Mirando dentro



Mirando dentro

NOTAS PARA EL ADULTO

- No es importante que el niño comprenda todos los conceptos expuestos en el capítulo precedente. Se trata simplemente de que le resulten familiares.

- Siga con el niño el recorrido que marcan las flechas del dibujo de la página 137. Dibuje en un papel o en la pizarra el esquema del ordenador y pida al niño que vaya dibujando las flechas siguiendo el orden secuencial del proceso.

- Los ordenadores funcionan exclusivamente con un código binario. Es decir, sólo admiten combinaciones de dos elementos (0 y 1). Así, en un BYTE, los 8 bits pueden combinarse de 256 maneras distintas. Efectivamente $2^8 = 256$. Por ejemplo:

  el número 30 vendría representado por 00011110

  el número 211 vendría representado por 11010011

- Existen ya en el mercado microprocesadores que son capaces de procesar 16 e incluso 32 BITS a la vez, lo que les confiere una extraordinaria capacidad operativa.

La Quinta operación

ARTURO se quedó muy pensativo cuando escuchó por primera vez eso de la quinta operación. El creía que las operaciones fundamentales eran cuatro: Sumar, restar, multiplicar y dividir. Pero hay una quinta operación, que es interesantísima y que permite hacer de manera muy fácil cálculos muy complicados.

Te vamos a poner unos ejemplos sencillos. Imagínate el número

1 000 000 000 000 000 000 000

Es equivalente al número de kilómetros que nos separan de algunas estrellas. Cuando los astrónomos tienen que hacer cálculos para mandar naves espaciales, tienen que operar muchas veces con estas cifras. Esto es muy pesado y puedes cometer muchos errores. No se puede trabajar bien con tantos ceros.

La Quinta operación

Una forma más simple para escribir el mismo número sería

$$10^{21}$$

que se lee "diez elevado a 21". Es decir, que el número 10 se multiplica por sí mismo 21 veces. Al número que nos indica las veces que multiplicamos un determinado número por sí mismo (en nuestro caso, el 21) se le llama exponente. Por eso esta operación se llama *exponenciación.*

Otro ejemplo más sencillo: El número 8 también se puede escribir así: 2^3. Esto quiere decir que hay que multiplicar el número 2 tres veces por sí mismo. Es decir:

$$2 * 2 * 2 = 8$$

★ Te vamos a contar ahora una historia muy curiosa:

Cuenta la leyenda que había en la INDIA un Rey que se encontraba espantosamente aburrido. Así que mandó llamar a todos los sabios de la corte para que inventaran un juego que le divirtiera. Muchos lo intentaron sin conseguirlo.

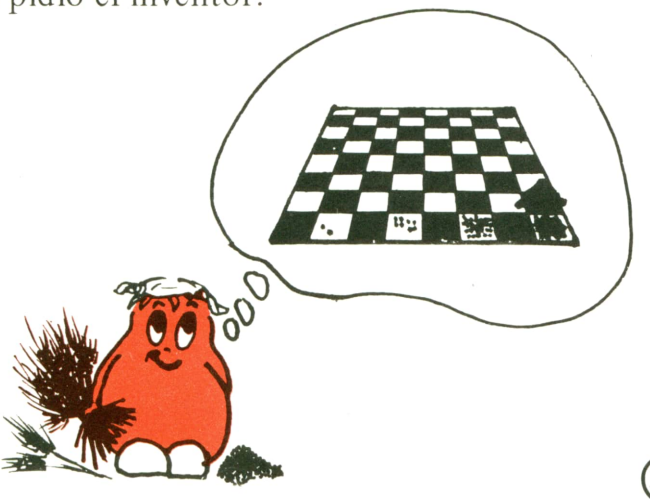
Hasta que un día apareció un personaje muy extraño que presentó al Rey un juego tan maravilloso, inteligente e interesante que el Rey quedó entusiasmado. Se trataba del juego que hoy conocemos como el juego del ajedrez.

La Quinta operación

El Rey agradecido, decidió premiar al inventor del ajedrez. Y le dijo: "Como premio a tu sabiduría, puedes pedirme lo que desees". El extraño personaje contestó: "Majestad, tan sólo quiero dos granos de trigo por la primera casilla del tablero, 4 granos por la segunda, 8 granos por la tercera, 16 granos por la cuarta.... y así sucesivamente hasta completar las 64 casillas del tablero de ajedrez".

El Rey se quedó muy contento, porque pensó que el inventor del ajedrez se conformaba con muy poco a cambio de un juego tan interesante, así que mandó llamar a su Primer Ministro y le ordenó que entregara al inventor lo que éste pedía.

Pero cuál fue su sorpresa cuando el Ministro vino a comunicarle que, aunque todo su pueblo se dedicase a cultivar trigo ... ¡no habría trigo en el mundo para reunir la cantidad que pidió el inventor!



La Quinta operación

Para que tú te hagas una idea, harían falta más o menos

18 446 744 500 000 000 000 granos de trigo

Esta cifra es tan grande, que casi ni la podemos imaginar. Si toda la tierra se dedicara a cultivar trigo, sin hacer otra cosa, se tardarían 45.000 años en reunir todo este trigo. Aunque corriéramos mucho, para contar todos los granos de trigo harían falta por lo menos 1.170 millones de siglos, o sea, 117.000 millones de años ¡INCREIBLE!

Eso es porque, como ya te habrás dado cuenta, por cada casilla el inventor pidió el doble de granos que le daban por la casilla anterior. Es decir, que multiplicaba por 2 el número de granos de la casilla anterior y así durante 64 casillas.

El enorme número anterior lo podríamos haber escrito 2^{64}. Parece mentira ¿eh?

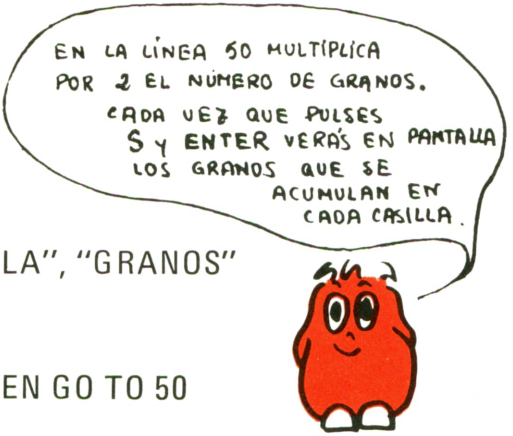
En un ordenador, 2^{64} se escribe así: 2 ↑ 64, porque en el ordenador el signo de la exponenciación es ↑.

★ Vamos a hacer un programa que reproduzca esta historia.

```
 5  REM "AJEDREZ"
10  PRINT " Pulsa S para otra casilla"
20  PRINT
```

La Quinta operación

```
30   LET C=0
40   LET A=1
50   LET A=A*2
60   LET C=C+1
70   PRINT "CASILLA", "GRANOS"
80   PRINT C, A
90   INPUT S$
100  IF S$="S" THEN GO TO 50
```



No hará falta que te expliquemos cómo funciona este programa pues todas las instrucciones son muy sencillas y ya las aprendiste en el libro BASIC PARA NIÑOS.

También podríamos haber escrito este programa utilizando un bucle FOR/NEXT.

```
5    REM "AJEDREZ"
10   PRINT "PULSA S PARA OTRA CASILLA"
20   PRINT
30   LET A=1
40   FOR C=1 TO 64
50   LET A= 2 ↑ C
60   PRINT "CASILLA", " GRANOS"
70   PRINT C, A
80   INPUT S$
90   IF S$="S" THEN GO TO 110
```

La Quinta operación

```
100  STOP
110  NEXT C
```

Ejecuta el programa. Verás como la cantidad de granos es cada vez mayor y, cuando llegues más o menos a la casilla 30, el ordenador escribe unas cifras un poco más "raras".

En la línea 30, por ejemplo, te aparecerá en pantalla:

1.07374182E + 9

Eso significa que el ordenador nos está diciendo *en su lenguaje* que también utiliza una forma simplificada para escribir un número grande.

El símbolo E + 9 es igual que decir que tienes que correr la coma 9 números hacia la derecha. Es decir, que si multiplicas 9 veces por 10 el número 1,07374182 obtendrás el número de granos que hay en la casilla 30 del tablero de ajedrez.

Casilla	*Expresión Matemática*	*Expresión Informática*	*Número*
30	2^{30}	2 ↑ 30 (Como tú lo escribirías) 1.07374182E +9 (Como te lo dice el computador)	1.073.741.820

La Quinta operación

Cuando llegues al final del programa, verás que el ordenador te da (utilizando su forma de expresarse) el número total de granos que corresponden a las 64 casillas.

El mismo resultado, pero sin verlo casilla a casilla, puedes verlo tecleando simplemente:

PRINT 2 ↑ 64

Como ves la exponenciación es muy interesante para manejar números grandes.

NOTAS PARA EL ADULTO

- Es muy probable que el niño no se sienta especialmente motivado hacia este tema. De ahí la importancia de explicárselo utilizando la leyenda del juego del ajedrez. Es probable que algunos niños no sean capaces de asimilar el concepto de magnitud numérica que se ha considerado en el ejemplo. No tiene importancia, como tampoco la tiene el que no comprendan del todo el concepto de exponencial.

La Quinta operación

Lo hemos incluido en este capítulo para que el niño lo conozca y no le resulte nuevo cuando mencionemos la operacion de exponenciación dentro del capítulo dedicado a los operadores aritméticos.

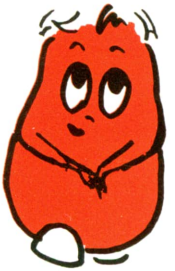
El desconocimiento del concepto exponencial no afectará en absoluto la comprensión de los temas que se tratarán a lo largo de este libro.

Haga ver al niño que la expresión gráfica que utilizará el ordenador para indicar una exponenciación es E + ... mientras que la que el niño debe utilizar a través del teclado es ↑.

No se han comentado los programas pues las instrucciones que aparecen en ellos son sobradamente conocidas por el niño. Si ha notado alguna dificultad de comprensión, repasen las instrucciones que se suponen ya conocidas.

Si a pesar de la utilización de los exponentes un número resulta demasiado grande para la capacidad del ordenador, éste nos dará un mensaje del tipo

?OV ERROR o NUMBER TOO BIG o similar.



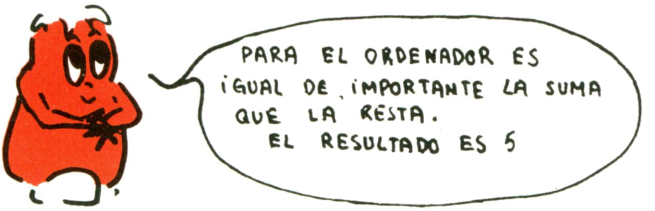
Operadores aritméticos

Ya sabes que la máquina es capaz de operar si tú se lo pides en una instrucción de tu programa. Cuando tú haces un programa, tienes mucho cuidado con el orden de las líneas, pues ése es el orden que leerá el ordenador.

Bien. Lo mismo tienes que hacer cuando quieras que el ordenador efectúe una operación. No puedes escribir las sumas, restas, multiplicaciones .... todo mezclado y sin orden. El ordenador es muy metódico y siempre sigue un mismo orden a la hora de operar.

★ Vamos a ver como operaría con la siguiente expresión:

PRINT 3 – 2 + 8 – 4



El ordenador sumará y restará los términos uno tras otro en el orden en que tú los has escrito.

Operadores aritméticos

Pero vamos a hacer un cambio:

PRINT 3 – 2 +8/4

Para el ordenador es mucho más importante la división que la suma o la resta, así que lo primero que hará es dividir los números a cada lado de la barra de dividir y luego sumará y restará.

★ Como has visto, existen unas reglas que el ordenador sigue. Son muy fáciles: Supongamos que tenemos un problema matemático en el que aparecen todas las operaciones (Suma, resta, multiplicación, división y exponenciación).

El ordenador

PRIMERO hace todas las exponenciaciones, de izquierda a derecha

SEGUNDO hace todas las multiplicaciones y divisiones, empezando por la izquierda

TERCERO hace todas las sumas y restas, empezando por la izquierda.

Operadores aritméticos

Pero cuando se encuentre con algo escrito entre paréntesis, hará primero todas las operaciones que estén dentro del paréntesis (y por el orden que él conoce) y luego seguirá actuando con las restantes operaciones siguiendo la regla que te hemos dicho antes.

★ Suponte que tú no quieres dividir 8 entre 4 y al resultado sumarle 3 y restarle 2, sino que quieres saber el resultado de 3 menos 2 mas 8 y todo eso dividirlo por 4.

Como tienes que indicarle cuál es el orden en que tú quieres que opere, escribirías:

```
PRINT (3 – 2 +8)/4
```

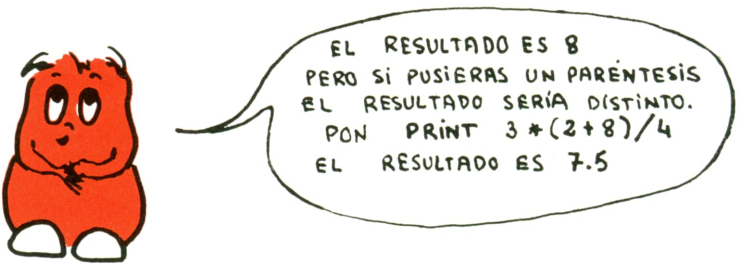
El ordenador hará primero siempre lo que se encuentre entre paréntesis ( ), así que el resultado ahora es 2.25.

★ Hagamos de nuevo otro cambio:

```
PRINT 3 * 2 +8 /4
```

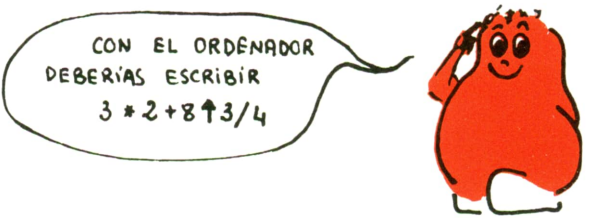
El ordenador multiplicará 3 por 2 y luego dividirá 8 entre 4. Al resultado de la multiplicación le sumará el resultado de la división. En este caso, el ordenador multiplica antes de dividir porque, en principio, el ordenador lee de izquierda a derecha.

Operadores aritméticos



★ Hagamos un nuevo cálculo:

PRINT $3 * 2 + 8^3/4$



Para el ordenador es más importante la exponenciación que la suma, multiplicación o división, por lo que primero calcula 8^3 (que es 512). Después, siguiendo el orden de izquierda a derecha, actúa así:

$\underbrace{3 * 2}$ + $\underbrace{512/4}$	Primero multiplica 3 por 2 y luego divide 512 entre 4.
6 + 128	Finalmente sumará los resultados obtenidos en las operaciones anteriores.

El resultado final es 134.

Operadores aritméticos

★ Otro ejemplo, ahora con paréntesis:

PRINT 3 * (2 + 8^3)/4

Primero eleva 8 a 3
Al resultado le suma 2
El resultado lo multiplica por 3
El resultado lo divide entre 4

Ten en cuenta que al escribir esto en tu ordenador lo harías así:

PRINT 3 * (2 + 8 ↑ 3)/4

Sigue paso a paso el orden de las operaciones y verás como esta vez el ordenador calcula primero lo que está entre paréntesis y luego sigue las reglas que ya sabemos.

★ ¿Qué pasaría si cambiamos el paréntesis de sitio? Escribe:

PRINT 3 * $(2 + 8)^3$/4

Primero suma 2 más 8
El resultado lo eleva a 3
El resultado lo multiplica por 3
El resultado lo divide entre 4

Operadores aritméticos

Como te habrás dado cuenta, dependiendo de los números y operaciones que encerremos dentro de los paréntesis, el resultado será distinto. Comprenderás la importancia que tiene que tengas muy claro las operaciones que quieres hacer antes de escribirlas de forma "informática". De lo contrario el ordenador te dará un resultado falso.

★ Fíjate en un ejemplo. ARTURO tenía 15 caramelos pero ha perdido 3 y los que le quedan tiene que repartirlos entre sus 6 amigos. Si escribe en el ordenador:

```
PRINT 15 – 3/6
```

el ordenador, siguiendo las reglas que conoces, daría como resultado 14.5. Como ves esto no puede ser. ARTURO debería escribir:

```
PRINT (15 – 3)/6
```

Veámoslo con un programa y utilizando además variables en lugar de números.

```
10 PRINT "CUANTOS CARAMELOS TIENE ARTURO?"
20 INPUT A
30 PRINT "CUANTOS PERDIO?"
40 INPUT B
```

Operadores aritméticos

```
50  PRINT "CUANTOS AMIGOS TIENE?"
60  INPUT C
70  LET X =( A– B)/C
80  PRINT " ARTURO DA "; X;"CARAMELOS A CA-
    DA UNO DE SUS"; C; "AMIGOS"
```

Fíjate en la importancia de escribir correctamente la línea 70. Cámbiala por:

```
70 LET X =A – B/C
```

y verás cómo el ordenador te da una respuesta absurda.

NOTAS PARA EL ADULTO

- La jerarquía de los operadores aritméticos es en realidad muy simple y fácil de comprender. Quizás resulte más difícil de explicar que de entender.

- El niño no debe retraerse ante la aparente profusión de operaciones. Son todas extraordinariamente sencillas y basta con practicar un poco para dominar el tema.

  Realicen conjuntamente distintas operaciones y cálculos en orden creciente de dificultad.

Operadores aritméticos

Haga ver al niño la importancia de conocer claramente a priori el cálculo que se desea realizar, a fin de que pueda transcribirlo correctamente en su expresión informática. Lo ideal es escribirlo previamente en un papel.

Comente con el niño la ventaja de escribir un programa como el que se incluye en este capítulo en lugar de teclear directamente PRINT (15 – 3)/6, lo cual evidentemente nos daría el mismo resultado. La utilización de programas con variables y la introducción de los valores a través de instrucciones INPUT permite la utilización de un mismo programa para diferentes casos.

Si lo considera oportuno o necesario, tenga en cuenta que es posible utilizar expresiones como ésta:

(3 * (2 + 8) ↑ 3)/4

en las que aparecen paréntesis externos e internos. El ordenador en estos casos dará siempre prioridad a los paréntesis internos.

CLEAR y CLS

Ya sabes que cada vez que empiezas un nuevo programa debes borrar todos los datos que el ordenador tiene en su memoria.

Generalmente habrás utilizado el comando **NEW**. Con ello el ordenador quedaba listo para volver a escribir un nuevo programa. Es decir, borrábamos de la memoria todos los datos y programas que el ordenador tenía memorizados.

Vamos a probar algunas cosas nuevas. Teclea este programa:

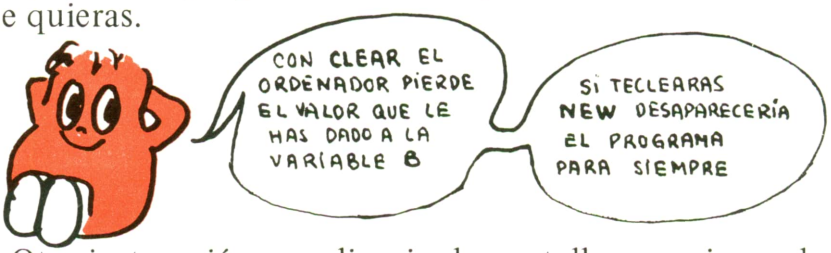
```
10  LET A=5
20  PRINT "DIME UN NUMERO"
30  INPUT B
40  LET M=A*B
50  PRINT "EL RESULTADO ES";M
```

Ahora tienes en la pantalla este pequeño programa. Teclea el comando **CLEAR** y verás qué pasa. El programa ha desaparecido de la pantalla.

CLEAR y CLS

Ahora teclea **LIST**. El programa vuelve a estar en pantalla. Esto ocurre porque el comando **CLEAR** no borra el programa.

Haz correr el programa y dale a B el valor 6, por ejemplo. En la pantalla podrás ver el resultado. Teclea **CLEAR** y luego lístalo. Fíjate en el programa. Está igual que al principio. A sigue valiendo 5 pero B no vale 6, sino que cuando ejecutes el programa otra vez, podrás darle a B el valor que quieras.



★ Otra instrucción para limpiar la pantalla pero sin perder el programa es **CLS**. Pero mientras que **NEW** y **CLEAR** eran comandos, es decir, órdenes directas sin línea de programa, **CLS** es una sentencia y debe ir incluida en una línea de programa. Cuando en la ejecución de un programa el ordenador llega a una línea con la instrucción **CLS**, borrará todo lo que en ese momento esté escrito en la pantalla. Añade al programa la línea:

```
45  CLS
```

Corre el programa y compara las diferencias.

CLEAR y CLS

Por eso es conveniente que al principio de cada programa la instrucción de tu primera línea sea **CLS**. Así tendrás de manera automática la pantalla limpia en cuanto lo ejecutes. Añade a nuestro programa: 5 CLS

NOTAS PARA EL ADULTO

Practique con el niño las sentencias y comandos explicados hasta conseguir su dominio.

En nuestro programa, B es una variable y A es una constante pues su valor lo hemos determinado nosotros mismos en la línea 10 del programa.

Tenga en cuenta que:

El comando **RUN** pone las variables a cero y borra o no la pantalla según el tipo de ordenador.

El comando **NEW** borra el programa de la memoria RAM.

El comando **CLEAR** borra la pantalla pero no la memoria, aunque pone las variables a cero.

La instrucción **CLS** borra la pantalla pero no altera las variables ni borra la memoria.

Conviene que a partir de ahora el niño se acostumbre a utilizar de forma sistemática la instrucción **CLS.**

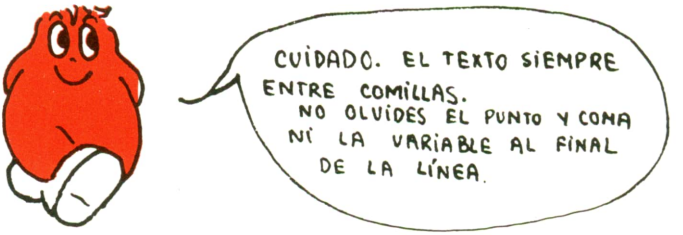
INPUT (Algo más)

Seguro que ya conoces para qué y cómo empleamos la instrucción **INPUT**. Hasta ahora habíamos combinado **PRINT** e **INPUT** para poder "hablar" con el ordenador. Vamos a ver ahora cómo **INPUT** nos permite ahorrarnos líneas de programa. Teclea de nuevo el programa de la página 41. Ahora cambia la línea 20 y escribe:

```
20  INPUT "ESTOY ESPERANDO UN NUMERO"; B
```

Borra la línea 30. Al ejecutar el programa te sale en pantalla exactamente lo mismo que antes. Has combinado en una sola línea las funciones que harían **PRINT** e **INPUT**. El programa completo te quedaría así:

```
 5  CLS
10  LET A=5
20  INPUT "ESTOY ESPERANDO UN NUMERO"; B
30  LET M=A * B
40  PRINT "EL RESULTADO ES"; M
```



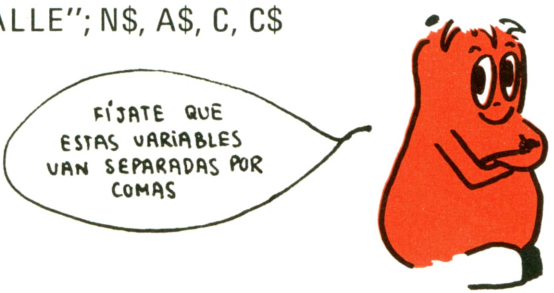
INPUT (Algo más)

★ Vamos a hacer un programa para practicar esta forma de utilizar **INPUT**. Ayudemos a ARTURO a completar su ficha de curso:

```
 5  CLS
10  INPUT " NOMBRE"; N$
20  INPUT "APELLIDO"; A$
25  INPUT "CURSO"; C
30  INPUT " CALLE"; C$
40  CLS
50  PRINT N$; " "; A$
60  PRINT "DE"; C; "CURSO"
70  PRINT "VIVE EN";  "  "; C$
80  END
```

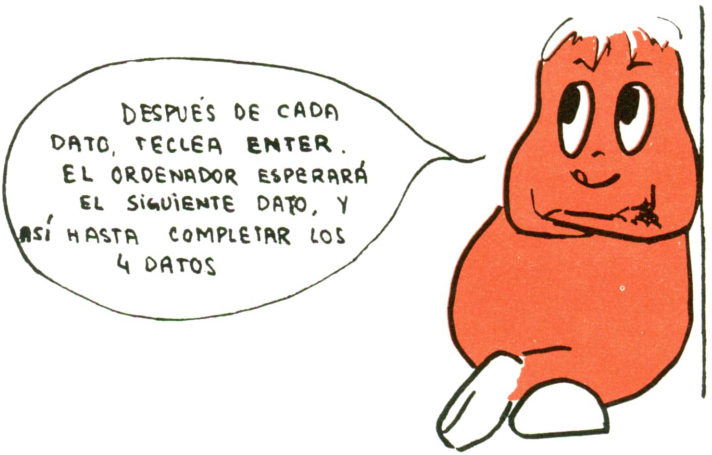
Pero todavía podemos hacer el programa más corto, pues **INPUT** nos permite que el ordenador nos pida más de una variable en una misma línea de programa. El programa anterior lo podríamos escribir:

```
 5  CLS
10  INPUT "DIME TU NOMBRE, APELLIDO, CURSO y
    CALLE"; N$, A$, C, C$
```



INPUT (Algo más)

```
15  CLS
20  PRINT "DATOS PERSONALES"
30  PRINT
40  PRINT N$; "  "; A$;" QUE ESTUDIA"; C; "CURSO
    VIVE EN  "; C$
```



Como ves, **INPUT** puede resultar muy cómodo. Observa una cosa curiosa de la línea 40. Entre N$ y A$ hemos dejado un espacio en blanco para que los textos asignados a cada una de estas variables queden bien separados.

Cuando en lugar de separar dos textos vayan a separar una variable numérica y un texto entre comillas, simplemente deja un espacio en blanco dentro de las comillas, como lo has venido haciendo hasta ahora.

INPUT (Algo más)

NOTAS PARA EL ADULTO

- Practique con el niño esta forma de utilizar **INPUT** y hágale ver que, dentro de una misma línea, puede hacer que el ordenador espere datos numéricos o alfanuméricos. Las variables deben ir separadas por comas.

- Practiquen igualmente la colocación de espacios dentro de las sentencias **PRINT,** a fin de evitar que se junten los textos. No todos los ordenadores actúan igual en este sentido pues algunos de ellos dejan automáticamente espacios en blanco a cada lado de un valor o dato numérico.

FOR/NEXT (Algo más)

Trabajaremos un poco sobre esta instrucción, que tú ya conoces. Usar **FOR/NEXT** en un programa es tan útil y tan cómodo, que conviene que te acostumbres a ella y la practiques mucho.

A partir de ahora, vamos a utilizarla con frecuencia, así que antes vamos a repasar un poco cómo funciona esta instrucción y de paso aprenderemos algunos trucos para aprovecharla mejor.

Teclea el programa:

```
10  FOR X=1 TO 3
20  PRINT X
30  NEXT X
```

Este programa es tan sencillo, que no hace falta que te lo expliquemos. Puedes escribirlo también así:

```
10  FOR X=1 TO 3 : PRINT X : NEXT X
```

FOR/NEXT (Algo más)

Por si no lo sabías, te diremos que dentro de una misma línea de programa se le pueden dar varias instrucciones al ordenador. Lo único que hay que hacer es separar una instrucción de otra por dos puntos (:). Esto lo puedes hacer con cualquier instrucción.

★ Una forma más completa de escribir **FOR/NEXT** sería:

```
10  FOR X = 1 TO 10 STEP 2
```

Esto quiere decir que X tomará valores de 1 hasta 10 pero de dos en dos. Si pusieras **STEP** 3 los tomaría de tres en tres y así sucesivamente.

★ Como otras instrucciones en BASIC, **FOR/NEXT** permite muchas combinaciones con otras instrucciones para facilitar la programación. Por ejemplo, escribe:

```
10  INPUT A
20  INPUT B
30  INPUT C
40  FOR X = A TO B STEP C
50  PRINT X
60  NEXT X
```

FOR/NEXT (Algo más)

Corre el programa y dale a las variables A, B y C el valor que tú quieras; por ejemplo, 0, 20 y 3. Tendrás una lista de números que empezará en 0 y escribirá valores de tres en tres. Fíjate que el último valor que escribe es 18. De acuerdo a la instrucción **STEP,** el próximo valor a escribir sería 21, pero tú le has dicho al ordenador que el último valor de X será 20, así que después de 18 ya no toma más valores.

★ Otra cosa que hay que tener en cuenta es que, si lo necesitas, puedes decirle al ordenador que se salga del bucle **FOR/NEXT**. Añade al programa anterior:

```
45   IF X=9 THEN GOTO 100
100  PRINT "ME HE SALIDO DEL BUCLE PORQUE
           X HA TOMADO EL VALOR 9"
```

Es decir, que con instrucciones del tipo **GO TO** o **IF .... THEN** puedes salir del bucle. Con ello el bucle queda interrumpido. Lo que no debes hacer es intentar que el ordenador entre en un bucle desde fuera del bucle, porque te surgirá algún error.

★ Sigamos viendo formas de utilizar **FOR/NEXT**. Generalmente, dentro de un bucle tú metes una serie de instrucciones que quieres que el ordenador te repita un número de veces determinado.

FOR/NEXT (Algo más)

Pues .... no pasa nada. Es decir, que tendríamos un bucle vacío. ARTURO no entendía muy bien para qué podíamos querer nosotros un bucle vacío, que no hiciera nada. La verdad es que tiene su utilidad. Aunque un bucle esté vacío, mientras lo está ejecutando el ordenador no hace otra cosa. Así que podemos utilizar un bucle así para retardar una acción del programa. Por ejemplo, teclea:

```
5  CLS
10 PRINT "ESTA LINEA SE BORRARA DENTRO DE
   UN MOMENTO"
20 CLS
30 PRINT "YA SE HA BORRADO"
```

Cuando ejecutes el programa, no te dará tiempo a ver en la pantalla el primer texto porque el ordenador, después de escribirlo en pantalla, sigue leyendo la línea 20 y te lo borra. Conviene que introduzcas en este programa un bucle vacío para retrasar la lectura de la línea 20 por el ordenador.

FOR/NEXT (Algo más)

Añade:

```
12  FOR X=1 TO 3000
13  NEXT X
```

Corre el programa. ¡Perfecto! Estos bucles se llaman también *bucles de retardo.* Puedes controlar el tiempo de retardo cambiando la instrucción **FOR.** Por ejemplo FOR X = 1 TO 1000, o lo que tú quieras.

Te recuerdo que en lugar de las líneas 12 y 13 podrías haber escrito una sola línea:

```
12  FOR X=1 TO 3000 : NEXT X
```

★ Vamos a ver otro ejemplo:

```
10  FOR X=1 TO 10
20  PRINT "PROBANDO BUCLES"
30  NEXT X
```

Al correr el programa, en pantalla te aparecerá 10 veces el texto de la línea 20.

FOR/NEXT (Algo más)

Añade ahora:

```
25  FOR Y=1 TO 2000
26  NEXT Y
```

Te aparecerá el mismo texto en pantalla, pero poco a poco, porque después de ejecutar la línea 20, el ordenador ejecuta el bucle de retardo antes de leer la línea 30 y tomar el siguiente valor de X.

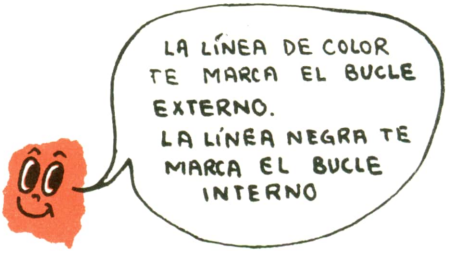
★ Casi sin darnos cuenta, hemos visto algo muy importante: Hemos metido un bucle dentro de otro bucle. Estos bucles se llaman bucles internos o, mejor, *bucles anidados.* ¿Por qué anidados? Porque el bucle de fuera es como un nido *dentro* del cuál está metido el bucle interno.

Y otra cosa muy importante: En nuestro ejemplo, el ordenador ejecuta primero todo el bucle de retardo (el interno) *antes* de tomar el siguiente valor del bucle de fuera o externo.

FOR/NEXT (Algo más)

★ ¿Qué pasaría si el bucle anidado no fuera un bucle vacío sino un bucle normal? Vamos a verlo con un ejemplo. Teclea el programa:

```
10  FOR A = 1 TO 2
20  FOR B = 1 TO 4
30  PRINT A, B
40  NEXT B
50  NEXT A
```



El bucle de fuera (externo) es el de las líneas 10 y 50. El bucle anidado (interno) es el de las líneas 20, 30 y 40. Fíjate en el importante detalle de que primero pones **NEXT** B (para cerrar el bucle interno) y luego **NEXT** A (para cerrar el bucle externo).

No te equivoques haciéndolo al revés, pues el programa no saldría. Veamos como funciona el programa. Sigue despacio y atentamente las explicaciones.

En la línea 10 le da el primer valor a A (A = 1).

En la línea 20 le da el primer valor a B (B = 1) y empieza a ejecutar el bucle anidado. A vale 1.

En la línea 30 escribe estos valores.

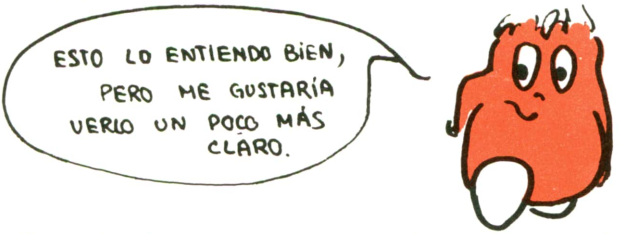
En la línea 40 le dices al ordenador que siga ejecutando el bucle interno y tome el siguiente valor de B (B = 2). En este momento A todavía vale 1.

FOR/NEXT (Algo más)

Como se está ejecutando el bucle interno, el ordenador vuelve a la línea 30 e imprime ahora los valores que tiene: (A = 1) y (B = 2).

De esta forma el ordenador sigue ejecutando el bucle interno hasta dar a la variable B todos los valores. Cuando B = 4, el ordenador ha terminado el bucle interno y llega a la línea 50, donde toma el siguiente valor de A. Ahora A vale 2.

Entonces, con este valor, vuelve a repetir todo el proceso del bucle interno. Así una y otra vez hasta terminar con todos los valores de A.



Para verlo más claro, ¿por qué no le pedimos al ordenador que nos ayude? Cambia la línea 30:

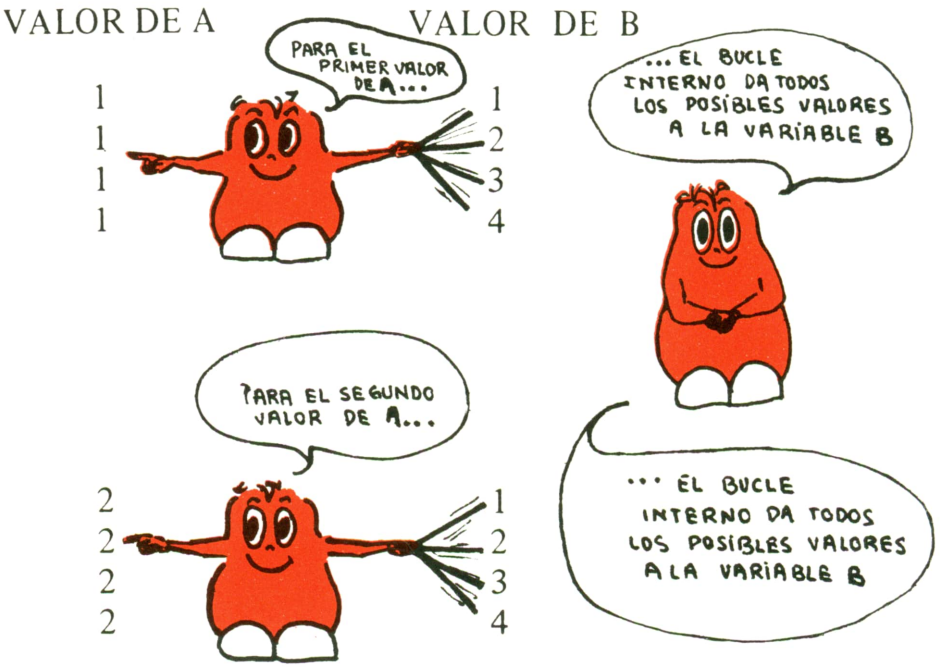
```
30  PRINT " A VALE"; A; "... AHORA B VALE"; B
```

Añade la línea:

```
35  IF B =4 THEN PRINT "TERMINADO EL BUCLE
    INTERNO PARA EL VALOR"; A; "DE LA VARIA-
    BLE A"
```

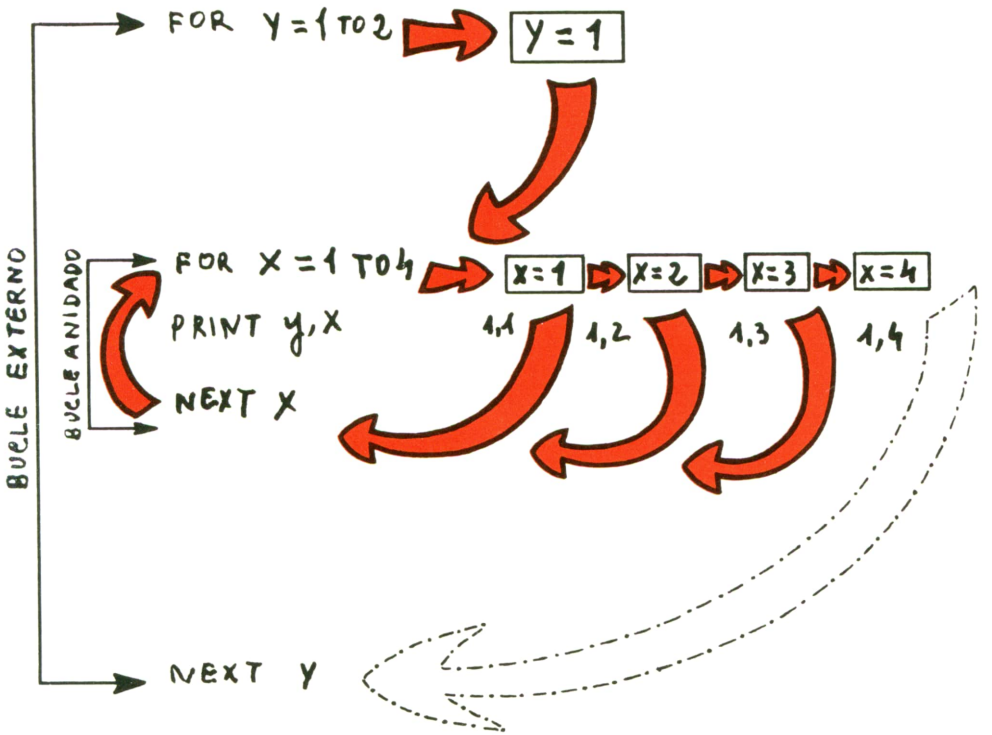
FOR/NEXT (Algo más)

Corre el programa. En pantalla tienes los distintos valores que van tomando uno detrás de otro las variables A y B.



Fíjate ahora en el dibujo de la página siguiente. Verás el recorrido que hace el ordenador cuando encuentra un bucle anidado.

FOR/NEXT (Algo más)



Vuelve a leer las explicaciones mirando la pantalla y el programa. Comprobarás que es fácil de entender.

★ Veamos algunos programas prácticos donde aplicar los bucles anidados. Por ejemplo, para obtener una lista de las distintas formas de leer las fichas de dominó, teclea:

FOR/NEXT (Algo más)

```
10  FOR X=0 TO 6
20  FOR Y=0 TO 6
30  PRINT X;" ! ";Y
40  NEXT Y
50  NEXT X
```

Para evitar que te aparezcan los números demasiado deprisa en la pantalla, añade el bucle de retardo:

```
35  FOR A=1 TO 500 : NEXT A
```

y los verás aparecer uno a uno.

★ Una forma de tener la tabla de multiplicar sería:

```
10  CLS
20  PRINT "TABLA DE MULTIPLICAR"
30  PRINT "PULSA ENTER DESPUES DE CADA MUL-
    TIPLICACION"
40  FOR A=1 TO 2500 : NEXT A : CLS
50  FOR X=1 TO 10
60  FOR Y=1 TO 10
70  LET C=X * Y
80  PRINT X;"POR";Y;"ES";C
90  INPUT E$
100 CLS
110 NEXT Y
120 NEXT X
```

CUANDO EL ORDENADOR LLEGA A LA LÍNEA 90 ESPERA QUE TECLEES ENTER PARA SEGUIR CON EL PROGRAMA

FOR/NEXT (Algo más)

Fíjate como en la línea 40 introducimos un bucle de retardo para retrasar el borrado de la pantalla. El bucle externo se indica en las líneas 50 y 120. El bucle anidado se indica en las líneas 60 y 110.

Atención al truco de la línea 90. Con ella obligamos al ordenador a pararse esperando un dato. Así evitamos que la pantalla se nos llene con la tabla de multiplicar. En cuanto le introducimos el dato (da igual el que sea, basta con pulsar **ENTER**), el ordenador sigue con el programa. En el capítulo de **"CADENAS..."** veremos una forma más correcta de conseguir el mismo efecto utilizando **INKEY$**.

Ejecuta el programa.

Bueno, por ahora es suficiente. Como ves, se pueden hacer muchas cosas con los bucles. Incluso, organizar una liga de baloncesto con los amigos de ARTURO. ¿Te atreverías a ayudarle? Si quieres ver cómo se hace, mira el programa de la página 260.

FOR/NEXT (Algo más)

NOTAS PARA EL ADULTO

Es conveniente que el niño practique esta instrucción, por su extraordinaria versatilidad y su constante utilización en BASIC.

Hágale ver que se pueden introducir infinitos bucles anidados dentro de un programa, pero que es imprescindible una adecuada colocación de las sentencias **NEXT,** pues de lo contrario los bucles se cruzarían, lo cuál daría error.

CORRECTO

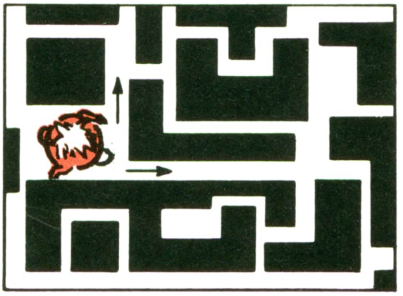
```
10  FOR A=0 TO 10
20  - - - - - - - -
30  - - - - - - - -
40  FOR B =0 TO 5
50  - - - - - - - -
60  - - - - - - - -
70  FOR C =0 TO
    100 STEP 5
80  - - - - - - - -
90  - - - - - - - -
100 NEXT C
110 NEXT B
120 NEXT A
```

INCORRECTO

```
10  FOR A=0 TO 10
20  - - - - - - - -
30  - - - - - - - -
40  FOR B =0 TO 5
50  - - - - - - - -
60  - - - - - - - -
70  FOR C =0 TO
    100 STEP 5
80  - - - - - - - -
90  - - - - - - - -
100 NEXT A
110 NEXT B
120 NEXT C
```

En los bucles de retardo hay que tener en cuenta que no todos los ordenadores tienen la misma velocidad de ejecución, por lo que un bucle del tipo FOR X = 1 TO 1000, puede retrasar la ejecución del programa más o menos tiempo en función del tipo de ordenador.

Operadores lógicos



Tú recuerdas que utilizando **IF THEN** le poníamos unas condiciones al ordenador. Por ejemplo, decíamos cosas como éstas:

```
IF A=B THEN GO TO 1
IF M>5 THEN PRINT " CORRECTO"
IF A$<>"ARTURO" THEN STOP
```

Te habrás dado cuenta que en cada una de estas líneas de programa sólo le poníamos una condición al ordenador. El BASIC, sin embargo, te permite ponerle más de una condición, con lo que se aumentan las posibilidades de programación. Vamos a ver como se hace utilizando los operadores lógicos. Los más utilizados son:

AND	que significa	Y
OR	que significa	O

Operadores lógicos

★ Te vamos a poner algunos ejemplos para que veas cómo operan.

ARTURO tiene ganas de jugar en el jardín y su mamá le dice:

"SI TIENES UN BALON Y TIENES TIEMPO PUEDES JUGAR"

La mamá de ARTURO le ha puesto dos condiciones.

La 1ª condición es que tenga un balón ... ¿Se cumple?

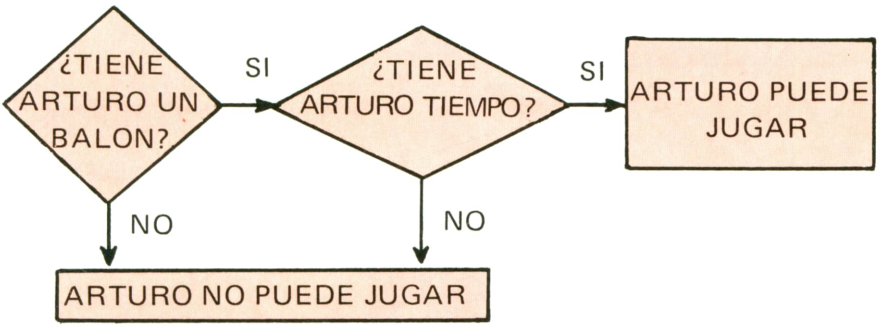
La 2ª condición es que tenga tiempo ... ¿Se cumple?

ARTURO tiene un balón y tiene tiempo para jugar. Como se cumplen las dos condiciones, ARTURO podrá salir a jugar. Pero como puede ocurrir que no siempre se cumplan las dos condiciones vamos a hacer una tabla para ver lo que puede pasar.

Operadores lógicos

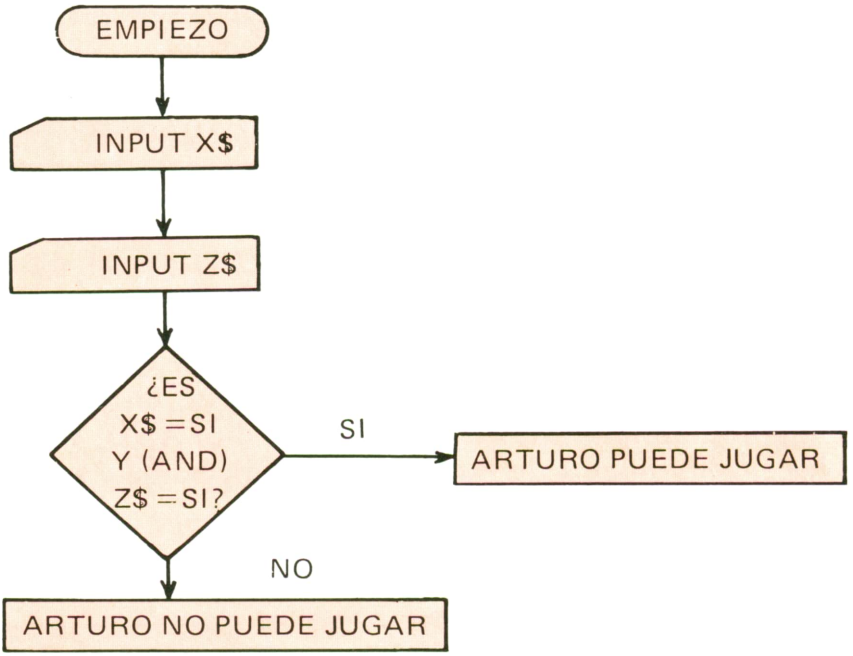
1ª condición Tener balón	*2ª condición Tener tiempo*	*Se cumple 1ª Y (AND) la 2ª?*	*Resultado*
SI	SI	SI	ARTURO saldrá
SI	NO	NO	ARTURO no saldrá
NO	SI	NO	ARTURO no saldrá
NO	NO	NO	ARTURO no saldrá

Como ves, solamente si ARTURO tiene un balón Y tiempo, saldrá a jugar. En BASIC, **Y** se dice **AND**. Vamos a hacer un esquema de esto y luego un diagrama y su programa para reproducir esta situación.



Operadores lógicos

El diagrama de flujo del programa sería:



Operadores lógicos

Y el programa sería:

```
 10  CLS
 20  PRINT "CONTESTA SI O NO A LAS PRE-
     GUNTAS"
 30  PRINT
 40  INPUT "ARTURO TIENE BALON?"; X$
 50  INPUT "ARTURO TIENE TIEMPO?"; Z$
 60  IF X$ = "SI" AND Z$ = "SI" THEN 120
 70  PRINT
 80  PRINT "COMO NO SE CUMPLEN"
 90  PRINT "LAS DOS CONDICIONES"
100  PRINT "ARTURO NO PUEDE JUGAR"
110  GOTO 30
120  PRINT
130  PRINT "COMO SE CUMPLEN"
140  PRINT "  LAS DOS CONDICIONES"
150  PRINT "  ARTURO PUEDE JUGAR"
155  PRINT "-----------------"
160  GOTO 30
```

Fíjate bien en la línea 60. Le estás diciendo al ordenador que sólo en el caso de que X$ y Z$ sean iguales a "SI"

vaya a la línea 120, a partir de la cual el ordenador nos reconfirma que por cumplirse las dos condiciones ARTURO puede ir a jugar. Ejecuta el programa y contesta como quieras a las preguntas que te hará el ordenador. Compara los resultados con la tabla de la página 177.

★ ¡Ah! Hay algo que te va a ayudar mucho cuando ejecutes un programa. Puede suceder que cuando teclees la respuesta a una pregunta que te haga el ordenador cometas un error. Por ejemplo, puede ocurrir que en el programa anterior teclees "SE" o "SU" u otra cosa en lugar de "SI".

Estos errores pueden afectar de manera importante al resultado del programa. Compruébalo contestando, por ejemplo "SI" a la primera pregunta y "SU" a la segunda. Verás como el ordenador te da una respuesta equivocada para tí, pues el ordenador solamente entiende que no has tecleado "SI" y lo entiende como "NO". Esto lo conoces tú ya perfectamente por la forma en que está escrito el programa en la línea 60 y siguientes: Si las dos respuestas no son exactamente "SI" y "SI", el ordenador sigue leyendo el programa en lugar de saltar a la línea 120.

Operadores lógicos

Así que es muy conveniente introducir en el programa una línea "chivata" que nos descubra cuándo hay un error de este tipo.

En nuestro último programa podemos utilizar precisamente una línea "chivata" que contiene el operador lógico **AND**, aunque en este caso no lo usamos para nada en la solución del problema de ARTURO sobre jugar o no en el jardín, sino únicamente para descubrir un posible fallo al pulsar las teclas. Añade:

```
 55  IF X$ <> "SI" AND X$ <> "NO" OR Z$ <> "SI"
     AND Z$ <> "NO" THEN 165
165  PRINT
170  PRINT "TECLEASTE MAL TU RESPUESTA. CON-
     TESTA OTRA VEZ"
180  GOTO 30
```

La línea 55 significa que si X$ es distinta de "SI" **y** distinta de "NO" **o** Z$ es distinta de "SI" **y** distinta de "NO" tu respuesta es incorrecta pues el programa te pedía contestar solamente "SI" o "NO". Fíjate que en esta línea utilizamos un nuevo operador lógico: **OR** (en español significa o).

Vamos a conocerlo mejor con un ejemplo, como hicimos antes. La mamá de ARTURO le dice ahora:

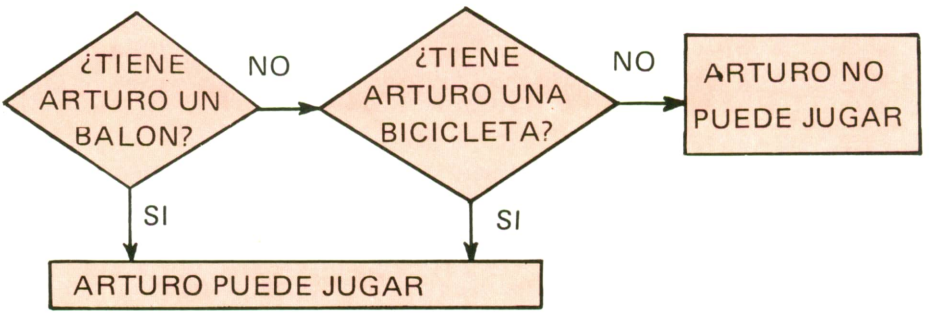
"SI TIENES UN BALON O TIENES UNA BICICLETA PUEDES JUGAR"

Operadores lógicos

La mamá de ARTURO ha puesto dos condiciones, pero como ves, basta que se cumpla una de las dos para que ARTURO pueda jugar. Hagamos nuestra tabla para ver las posibilidades que tiene ARTURO de salir a jugar.

1ª Condición Tener balón	*2ª Condición Tener bicicleta*	*Se cumple 1 o (OR) la 2ª*	*Resultado*
SI	SI	SI	ARTURO jugará
SI	NO	SI	ARTURO jugará
NO	SI	SI	ARTURO jugará
NO	NO	NO	ARTURO no jugará

Como ves, basta que ARTURO tenga una de las dos cosas para que salga a jugar. En BASIC, O se escribe **OR**. El esquema de lo que puede suceder sería así:



Operadores lógicos

Puedes hacer también un programa para ver las posibilidades de ARTURO. Sobre la base del programa anterior, cambia las líneas siguientes:

```
 50  INPUT "ARTURO TIENE BIBICLETA?"; Z$
 60  IF X$ = "SI" OR Z$ = "SI" THEN 120
 85  PRINT "NINGUNA DE"
130  PRINT "COMO SE CUMPLE UNA DE "
```

Ahora en la línea 60 le estás diciendo al ordenador que si se cumple que X$ o Z$ son iguales a "SI" vaya a la línea 120, a partir de la cual el ordenador nos reconfirma que por cumplirse una de las dos condiciones ARTURO puede ir a jugar. Ejecuta el programa y compara los resultados con los de la tabla de la página 182. Fíjate que aquí seguimos utilizando la línea 55 (la "chivata") igual que estaba en el programa anterior.

Estos ejemplos te habrán servido para entender mejor las tablas y el funcionamiento de los operadores lógicos. Estos se utilizan con cierta frecuencia y, cuando te veas en la necesidad de usarlos, te parecerán muy sencillos.

★ Te vamos a proponer una aplicación muy sencilla y muy .... misteriosa.

Operadores lógicos

ARTURO cree que es un espía y tiene un montón de programas con datos secretos que no quiere que nadie vea. Así que ha decidido poner una clave secreta a cada programa para que nadie que no conozca la clave pueda ejecutar el programa. Tú puedes hacer lo mismo, poniendo delante de cada programa que hagas unas líneas como éstas:

```
1   LET C=0
2   CLS: PRINT "ESTE PROGRAMA ES SECRETO.
    CONTESTE CORRECTAMENTE A LAS CLAVES":
    PRINT
3   INPUT "DIGA EL NUMERO Y DESPUES LA PALA-
    BRA SECRETA"; M, L$
4   LET C=C+1: IF C=3 THEN 7
5   IF M=25 AND L$="ARBOL" THEN 10
6   GOTO 2
7   PRINT "  USTED NO CONOCE LA CLAVE. DEBE
          SER UN ESPIA. TUVO TRES OPORTUNI-
          DADES. ESTE PROGRAMA SE AUTODES-
          TRUIRA EN 5 SEGUNDOS"
8   FOR Y=1 TO 3000: NEXT Y
9   CLS: STOP
10  ......(aquí seguirá tu programa)
```

Operadores lógicos

El contador C de la línea 4 controlará el número de intentos. Cuando C toma el valor 3, el ordenador salta a la línea 7, donde, después de imprimir un texto de aviso, borra el programa de la pantalla. Fíjate en el bucle de retardo de la línea 8. En la línea 9, puedes sustituir la instrucción **STOP** por **NEW**, pero en ese caso el programa entero se borrará de la memoria.

Haz tu propia clave secreta cambiando a tu gusto los valores de M y L$ en la línea 5.

Hagamos otro cambio. Sustituye en la línea 5 **AND** por **OR** y corre el programa. Comprobarás que en este caso basta con que aciertes una de las dos claves para que el ordenador te permita seguir la ejecución del programa principal.

¡Ah! En la línea 5 hemos puesto al final **THEN** 10 porque suponemos que tu programa principal empieza en la línea 10. Haz los cambios necesarios si tu programa empieza en otra línea.

Operadores lógicos

NOTAS PARA EL ADULTO

- Los operadores lógicos sólo actúan entre dos posibilidades: verdadero (representado por 1) y falso (representado por Ø). En el análisis de las posibilidades que nos ofrecen los operadores lógicos se utilizan las llamadas TABLAS DE VERDAD. La tabla correspondiente al operador lógico **AND** sería:

X	Y	X AND Y
1	1	1
0	0	0
0	1	0
1	0	0

y la tabla del operador lógico **OR** sería:

X	Y	X OR Y
1	1	1
0	0	0
0	1	1
1	0	1

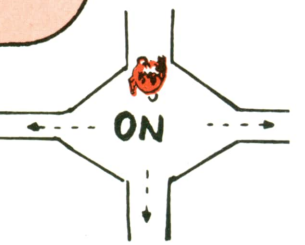
Operadores lógicos

Estos conceptos tienen frecuentes aplicaciones no sólo en Informática, sino también en Lógica y en Lógica Matemática. Conviene que el niño los asimile, lo cual es muy sencillo si se ejemplifica la teoría con planteamientos como los expuestos en este capítulo.

Acomode los ejemplos a la capacidad e interés del niño.

Existen otros operadores lógicos (**NOT, XOR, NOR...**) cuya función es más compleja y que no han sido analizados por considerarlos de un nivel conceptual superior a la capacidad del niño al que está destinado este libro.

IF/THEN (Algo más)



Vamos a mostrarte lo que es un MENU y las distintas formas que puede adoptar la instrucción **IF/THEN**.

Vamos a ver cómo funciona con un ejemplo. Teclea el programa:

```
 10  CLS
 20  INPUT "DI UN NUMERO"; A
 30  INPUT "DI UN NUMERO"; B
 40  PRINT "PULSA 1 PARA MULTIPLICAR"
 50  PRINT "PULSA 2 PARA DIVIDIR"
 60  PRINT "PULSA 3 PARA SUMAR"
 70  PRINT "PULSA 4 PARA OPERAR CON OTROS NU-
     MEROS"
 80  PRINT "PULSA 5 PARA TERMINAR"
 90  INPUT "QUE OPCION QUIERES?"; X
100  IF X <> 1 AND X <> 2 AND X <> 3 AND X <> 4
     AND X <> 5 THEN 220
110  IF X = 1 THEN 160
120  IF X = 2 THEN 180
130  IF X = 3 THEN 200
140  IF X = 4 THEN 10
```

IF/THEN (Algo más)

```
150  IF X = 5 THEN 230
160  PRINT "EL RESULTADO DE MULTIPLICAR "; A;
     "POR"; B; "ES"; A * B
170  GO TO 90
180  PRINT "EL RESULTADO DE DIVIDIR"; A; "EN-
     TRE"; B; "ES"; A/B
190  GO TO 90
200  PRINT "EL RESULTADO DE SUMAR"; A; "MAS";
     B; "ES"; A + B
210  GO TO 90
220  "TECLEASTE MAL TU OPCION. TECLEALA DE
     NUEVO" : GO TO 90
230  STOP
```

Como ves es un programa muy sencillo. En las líneas 40 hasta la 90 estás escribiendo lo que se llama un MENU.

Bueno, no vamos a comer. Pero al igual que en un restaurante te dan un menú para que elijas la comida que te

gusta, en un programa puedes poner al principio un menú para que al ejecutarlo puedas elegir la opción que más te interesa.

De esta manera tienes un programa más claro y más fácil de interpretar. Pon un menú en todos los programas que hagas y que permitan elegir entre varias opciones. ¡Ah! No te olvides de poner la línea "chivata" para evitar errores.

Fíjate en la línea 160. El resultado de multiplicar A por B no lo hemos asignado a ninguna variable, sino que le hemos dicho al ordenador que escriba directamente A * B. Igual hacemos con las líneas 180 y 200.

★ Bueno, ahora vamos a utilizar **ON GOTO**. No todos los ordenadores te permiten utilizar esta instrucción, así que consulta primero en tu manual si puedes emplearla o no.

Borra las líneas 110, 120, 130, 140 y 150 y escribe:

```
110 ON X GO TO 160, 180, 200, 10, 230
```

Verás que el resultado es el mismo al correr el programa. Con **ON GOTO** es como si le dijeras al ordenador "dependiendo de lo que valga X te irás a las líneas...."

X valdrá siempre 1, 2, 3... y así hasta el número de opciones posibles que queramos incluir en el programa.

Otra posibilidad que te ofrece **ON** es combinarlo con subrutinas. Es decir, utilizando la forma **ON X GOSUB**...

IF/THEN (Algo más)

NOTAS PARA EL ADULTO

La instrucción **ON GOTO** o **ON GOSUB** no es fundamental, sino simplemente un auxiliar de la programación. El niño puede practicar con ella para adquirir soltura y velocidad de programación.

Tenga en cuenta que no todos los ordenadores disponen de esta instrucción. Si el suyo no la tiene, no deje de comentar con el niño otros conceptos importantes desarrollados en este capítulo, como son la presentación de un programa con un menú (líneas 40 a 90), el empleo de líneas de detección de errores al introducir datos en la ejecución del programa (línea 100) y la impresión directa del resultado de operaciones aritméticas sin asignación previa a una determinada variable (líneas 160, 180 y 200).

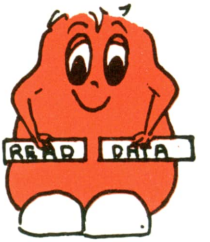
Si comprueba que el niño asimila con relativa facilidad estos conceptos puede ampliar sus conocimientos relativos a la instrucción **IF...THEN** explicándosela en su forma más completa:

IF (Condición) **THEN** (Instrucción) **ELSE** (Instrucción)

Es decir, si la condición se cumple, se ejecutará la instrucción que sigue a **THEN.** Si no se cumple, se ejecutará la instrucción que sigue a **ELSE.** Por ejemplo:

```
IF A = 5 THEN PRINT "ACERTASTE" ELSE
PRINT "FALLASTE"
```

READ y DATA



ARTURO va a organizar unas MINI OLIMPIADAS, y quiere saber con qué amigos cuenta para cada prueba. Para tenerlo todo muy bien organizado quiere llevar estos datos con ayuda del ordenador.

Para poder hacerlo cómodamente, va a escribir un programa utilizando la instrucción **READ/DATA**. ¿Le ayudamos? Vamos a seguir paso a paso la manera de hacerlo.

ARTURO tiene amigos especialistas en tres pruebas: CARRERA, BALONCESTO y NATACION. Así que lo primero que hace es preparar unas sentencias **DATA** con el nombre de sus amigos y la especialidad que practican:

```
1   DATA LUIS, CARRERA
2   DATA JUAN, BALONCESTO
3   DATA CARLOS, NATACION
4   DATA JOSE, BALONCESTO
5   DATA PEDRO, BALONCESTO
6   DATA JORGE, CARRERA
7   DATA MIGUEL, NATACION
8   DATA ARTURO, BALONCESTO
```

READ y DATA

Estos datos serán leídos después por una instrucción **READ.** Pero eso ya lo veremos más adelante. Ahora vamos a hacer un Menú, como tú ya conoces:

```
10  CLS
20  PRINT "PULSA 1 PARA CORREDORES"
30  PRINT "PULSA 2 PARA BALONCESTISTAS"
40  PRINT "PULSA 3 PARA NADADORES"
50  PRINT "PULSA 4 PARA FIN DE PROGRAMA"
60  PRINT: INPUT "TECLEA TU OPCION"; X
70  IF X >4 OR X <1 THEN PRINT "TECLEASTE
    MAL TU OPCION": GOTO 60
```

Fíjate la forma tan sencilla y original de detectar errores de tecleado en la línea 70 (la "chivata").

En las líneas 80, 90 y 100 le diremos al ordenador que se vaya a las diferentes subrutinas del programa según la opción que elijas:

```
 80  IF X =1 THEN GOSUB 1000
 90  IF X =2 THEN GOSUB 2000
100  IF X =3 THEN GOSUB 3000
110  IF X =4 THEN PRINT "FIN DE PROGRAMA":
     GOTO 200
200  END
```

READ y DATA

Escribamos la subrutina para buscar a los corredores:

```
1000  READ A$, B$
1010  IF B$="CARRERA" THEN 1030
1020  GO TO 1000
1030  PRINT A$
1040  GOTO 1000
```

Analicemos como funciona esta subrutina. En la línea 1000 se leen los datos almacenados en las líneas **DATA** del principio del programa. En la línea 1010 le decimos al ordenador que cuando B$ sea igual a "CARRERA" vaya a la línea 1030 donde escribirá A$ (el nombre del amigo que tiene esa especialidad). Vuelve después a la línea 1000 donde continuará leyendo las sentencias **DATA** que le faltan por leer.

Si B$ es distinto de "CARRERA", el ordenador continúa leyendo el programa y en la línea 1020 recibirá la orden de volver a la línea 1000 para seguir buscando datos.

Añade también:

```
150  GOTO 20
```

Así te evitas tener que teclear RUN cada vez que quieres organizar otra búsqueda.

READ y DATA

Ejecuta el programa como está. Pero ¡cuidado! de momento sólo puedes elegir la opción 1, porque todavía no hemos escrito las subrutinas correspondientes a las otras opciones.

En la pantalla te aparecerán los amigos de ARTURO que son especialistas en CARRERA. También te aparecerá un mensaje indicándote que el ordenador no tiene más datos que leer desde la línea 1000. Por eso conviene que añadas un dato "falso":

```
9  DATA FIN, FIN
```

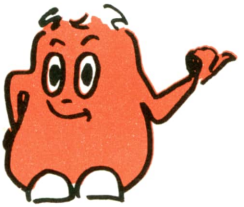
y en la subrutina añade la línea:

```
1015  IF B$ = "FIN" THEN PRINT "LOCALIZADOS LOS
      CORREDORES": RETURN
```

con lo cual damos la subrutina por terminada.

Con **RETURN** la primera línea que ejecutará el ordenador será la 150.

Hagamos ahora la subrutina que nos dará la lista de baloncestistas:



READ y DATA

```
2000  READ A$, B$
2010  IF B$ = "BALONCESTO" THEN 2030
2015  IF B$ = "FIN" THEN PRINT "LOCALIZADOS LOS
      BALONCESTISTAS": RETURN
2020  GOTO 2000
2030  PRINT A$
2040  GOTO 2000
```

No hace falta que te la expliquemos, porque funciona exactamente igual que la subrutina anterior.

Pero ¡ATENCION! Ejecuta ahora el programa. Pide la opción que tú quieras (la 1 o la 2). El ordenador te ofrece en pantalla la lista de los amigos cuya especialidad es la que acabas de pedir.

Ahora pide la otra opción. El ordenador no te imprime ningún nombre y te da un mensaje de error: No tiene datos para leer.

Pues, muy sencillo. Cuando una sentencia **READ** está leyendo valores almacenados en las sentencias **DATA**, es como si en cada dato colocara un punto o señal imaginario para saber que ese dato ya ha sido leído.

RESTORE

Cuando pides la segunda opción, y el ordenador lee los datos para volver a seleccionar, se encuentra que todos los datos están marcados con ese punto imaginario. Como para el ordenador ya están leídos, no los tiene en cuenta. Por eso te aparece el mensaje de que no tiene datos.

Esto se lo decimos con la instrucción **RESTORE**. Esta instrucción obliga al ordenador a leer otra vez todas las sentencias **DATA**.

Por eso, para que siempre que lo necesitemos pueda leer todos los **DATA**, pondremos la instrucción **RESTORE** en todas nuestras subrutinas. Añade las líneas:

```
900   RESTORE: CLS
1900  RESTORE: CLS
```

Cambia las líneas 80, 90 y 100 así:

```
80    IF X=1 THEN GOSUB 900
90    IF X=2 THEN GOSUB 1900
100   IF X=3 THEN GOSUB 2900
```

RESTORE

Añade finalmente la subrutina que nos dará los nadadores:

```
2900  RESTORE: CLS
3000  READ A$, B$
3010  IF B$ = "NATACION" THEN 3030
3015  IF B$ = "FIN" THEN PRINT "LOCALIZADOS LOS
      NADADORES" : RETURN
3020  GOTO 3000
3030  PRINT A$
3040  GOTO 3000
```

NOTAS PARA EL ADULTO

- Las instrucciones **READ, DATA** y **RESTORE** son muy útiles pues simplifican mucho el trabajo de asignación de valores a las variables. Conviene que el niño, ya en esta fase, utilice normalmente las mismas como un auxiliar más de la programación.

- Sigan el programa precedente paso a paso con el ordenador. No continúen con cada nueva fase sin haber entendido perfectamente el paso anterior.

RESTORE

Algunos ordenadores exigen que los datos incluidos en una sentencia **DATA** se escriban entre comillas. En otros no es necesario.

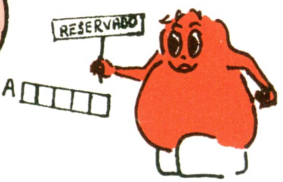
Presten atención a las líneas 1010, 2010 y 3010. No cometan ningún error ni incluyan espacios en blanco al escribir el texto entrecomillado que estamos asignando a las variables. De hacerlo así, al pedir una especialidad, el ordenador buscará en **DATA** un valor que no existe, pues el ordenador interpreta un espacio en blanco como un carácter más.

Siguiendo los mismos criterios ya expuestos y haciendo las modificaciones necesarias puede ampliar los campos añadiendo más especialidades, o más opciones (por ejemplo, la edad de los participantes). Si tiene dificultades, vea el programa tipo de la página 264.

Escribiendo **RESTORE** seguido de un número de línea que contenga una sentencia **DATA,** el ordenador leerá los datos existentes a partir de ese número de línea. Por ejemplo, en nuestro programa, un **RESTORE** 4 ignoraría a LUIS, JUAN y CARLOS en la segunda y sucesivas lecturas.

Los datos del programa se han incluido en las líneas 1 a 9 para una mayor claridad, pero igualmente pueden colocarse al final del mismo o, incluso, en la forma: **DATA** LUIS, CARRERA, JUAN, BALONCESTO, CARLOS, NATACION, JOSE ... etc.

DIM



Vamos a conocer una instrucción que evita mucho trabajo. Desde que aprendió a utilizarla, ARTURO aprovecha todas las ocasiones que puede y la emplea en sus programas. La verdad es que ahorra mucho tiempo.

¿Cómo descubrió ARTURO la intrucción **DIM**? Todo surgió un día que nuestro amigo quiso clasificar y ordenar algunas de sus cosas. Primero se hizo una relación:

LIBROS	**JUGUETES**	**ROPA**
PITUFOS	BALON	PANTALON
SUPERMAN	BICICLETA	CAMISA
GALAXIAS	RAQUETA	GORRA
INDIANA JONES	PATINES	ZAPATOS
PATO DONALD		PIJAMA
PEQUEÑA LULU		

Después decidió darle a cada objeto una variable. Así que empezó a escribir:

DIM

```
LET A$ = "PITUFOS"
LET B$ = "SUPERMAN"
.................................
.................................
```

ARTURO tiene razón. Va a tardar mucho tiempo en definir todas las variables. Además, como luego tenga que añadir más objetos, se le van a acabar las variables. Y si luego quiere que el ordenador le dé una relación de todos los objetos, clasificados por temas, tendrá que poner en su programa instrucciones del tipo **PRINT** A$, **PRINT** B$, **PRINT** C$, etc.

Así que decidió utilizar lo que se llama *variables con subíndice.* Una variable con subíndice se escribiría, por ejemplo:

A (1), A (2), A (3) ... etc.

Como ves, todas tienen la misma letra pero son variables distintas.

Trabajar con ellas tiene muchas ventajas. En primer lugar, la asignación de estas variables se puede hacer más rápidamente utilizando un bucle **FOR/NEXT**. Vamos a trabajar con los libros de ARTURO.

DIM

Escribe primero:

```
 5  CLS
10  DIM L$ (6)
```

Esto quiere decir que el ordenador va a reservar en su memoria 6 casillas para 6 variables que tendrán todas el mismo prefijo o la misma letra. En nuestro caso, L$. Así que las variables serán L$ (1), L$ (2), L$ (3), L$ (4), L$ (5) y L$ (6). Vamos a darles valores a estas variables. Como todas tienen el mismo prefijo, no es necesario que escribamos 6 líneas de programa con **LET**, sino que podemos utilizar un bucle **FOR/NEXT**. Escribe:

```
20  FOR N = 1 TO 6
25  PRINT "Escribe el libro"
30  INPUT L$ (N)
35  CLS
40  NEXT N
```

El bucle que empieza en la línea 20 va a hacer que N tome uno detrás de otro valores desde 1 hasta 6. Por 6 veces, el ordenador nos preguntará que valor le queremos asignar a L$ (N). Si sigues el orden que puso ARTURO al principio de este capítulo, las variables quedarán así:

```
L$ (1) = PITUFOS
L$ (2) = SUPERMAN
L$ (3) = GALAXIAS
L$ (4) = INDIANA JONES
L$ (5) = PATO DONALD
L$ (6) = PEQUEÑA LULU
```

El ordenador los tiene ya todos memorizados. Para probarlo, teclea, por ejemplo, **PRINT** L$ (3) o **PRINT** L$ (5). ¡El ordenador te contesta!

Así que podemos hacer que el ordenador nos imprima —diciéndoselo con una sola instrucción— el valor de todas las variables que tienen como prefijo común L$. Añade al programa las líneas:

```
50  FOR N = 1 TO 6
60  PRINT L$ (N)
70  NEXT N
```

Con este bucle el ordenador te imprimirá en pantalla todos los libros de ARTURO. Fíjate todo el trabajo y el tiempo que te has ahorrado.

Si queremos repetir el mismo proceso para los juguetes y para la ropa de ARTURO, lo mejor es utilizar un menú.

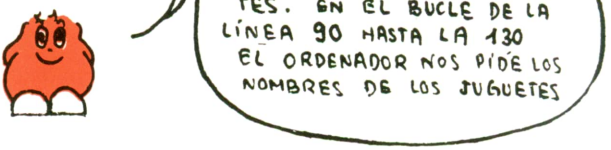
DIM

Te sugerimos que utilices el prefijo J$ para todas las variables de los juguetes y R$ para las variables de la ropa:

★ Vamos a hacer un programa para conocer los distintos objetos que tiene ARTURO clasificados por grupos. Empecemos desde el principio. Borra antes el programa anterior.

Primero prepararemos al ordenador para que reciba todos los datos:

```
 10  CLS
 20  DIM L$ (6)
 30  FOR N = 1 TO 6
 40  PRINT " ESCRIBE EL LIBRO"
 50  INPUT L$ (N)
 60  CLS
 70  NEXT N
 80  DIM J$ (4)
 90  FOR N = 1 TO 4
100  PRINT "ESCRIBE EL JUGUETE"
110  INPUT J$ (N)
120  CLS
130  NEXT N
```



DIM

```
140  DIM R$ (5)
150  FOR N = 1 TO 5
160  PRINT "ESCRIBE LA ROPA"
170  INPUT R$ (N)
180  CLS
190  NEXT N
```

Ahora escribiremos un menú para que en su momento podamos elegir cualquiera de las tres listas de objetos que tenemos.

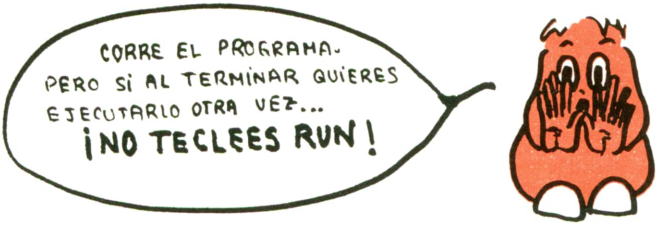
```
200  CLS
210  PRINT "OPCIONES:"
220  PRINT "PARA LIBROS PULSA 1"
230  PRINT " PARA JUGUETES PULSA 2"
240  PRINT " PARA ROPA PULSA 3"
250  INPUT " PULSA LA OPCION ELEGIDA "; X
260  IF X = 1 THEN 300
270  IF X = 2 THEN 360
280  IF X = 3 THEN 410
290  IF X <> 1 AND X <> 2 AND X <> 3 THEN PRINT
     " TECLEASTE MAL TU OPCION": GO TO 250
```

DIM

Finalmente, hagamos la parte del programa que nos escribirá la lista escogida.

```
300   CLS
310   FOR N=1 TO 6
320   PRINT L$ (N)
330   NEXT N
340   GO SUB 500
350   CLS: STOP
360   CLS
370   FOR N=1 TO 4
380   PRINT J$ (N)
390   NEXT N
400   GO SUB 500
410   CLS : STOP
420   CLS
430   FOR N=1 TO 5
440   PRINT R$ (N)
450   NEXT N
460   GO SUB 500
470   END
500   INPUT "PULSA K PARA OTRA LISTA. PULSA
      OTRA TECLA PARA TERMINAR"; K$
510   IF K$="  K" THEN 200
520   RETURN
```

DIM

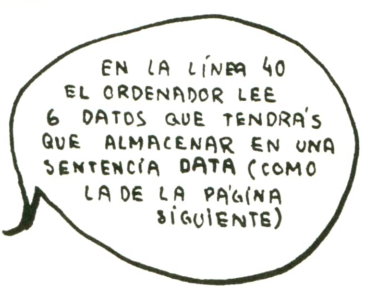


Este programa tiene un inconveniente: Cada vez que teclees **RUN** y corras el programa tendrás que introducir nuevamente los datos, pues como ya sabes, **RUN** pone todas las variables a cero.

Si tuvieras el programa guardado en un cassette con los datos ya introducidos, la única forma de evitar que se borraran sería tecleando **GOTO** 200 en lugar de **RUN**. En la línea 200 el programa empieza a pedirte opciones. Esto no es muy práctico.

★ Se puede hacer el programa de manera que el ordenador conserve en memoria todos los datos. Podemos usar las instrucciones **READ/DATA**. La estructura general del programa no cambia, pero, para introducir los datos en la memoria del ordenador escribiríamos:

```
10  CLS
20  DIM L$ (6)
30  FOR N = 1 TO 6
40  READ L$ (N)
50  NEXT N
```



DIM

Y en alguna parte del programa (preferentemente al final) pondrías una sentencia **DATA**:

```
800  DATA PITUFOS, SUPERMAN, GALAXIAS, INDIA-
     NA JONES, PATO DONALD, PEQUEÑA LULU
```

Siguiendo el mismo método tendrías que hacer cambios parecidos para los juguetes (líneas 80 hasta la 130) y para la ropa (líneas 140 hasta la 190). También tendrás que escribir dos líneas más de **DATA**, con los juguetes y la ropa.

NOTAS PARA EL ADULTO

La instrucción **DIM** es muy utilizada en la realización de listas y tablas. Conviene que el niño la comprenda perfectamente antes de abordar el estudio de capítulos posteriores, en los que se aplicará de forma habitual.

El ejemplo utilizado en este capítulo no resulta –por su estructura– verdaderamente práctico en la vida real, pero ayuda al niño a comprender y practicar insistentemente la instrucción **DIM** a lo largo del programa.

DIM

- En la mayoría de los ordenadores, sólo es necesaria la utilización de **DIM** cuando se van a utilizar más de 10 variables indexadas.

  En caso de utilizar más variables que las previamente dimensionadas se producirá un error, que su ordenador detectará inmediatamente.

- En nuestro programa se han empleado variables de caracteres a fin de aplicar un ejemplo más motivador para el niño. **DIM** puede utilizarse igualmente con variables numéricas.

  Cuando emplee variables alfanuméricas, debe tener en cuenta que algunos ordenadores exigen que al establecer la dimensión se especifique también el número máximo de caracteres que puede tener cualquiera de los valores asignados a las variables. Así, en nuestro ejemplo anterior la línea 20 se escribiría:

  ```
  20  DIM L$ (6, 13)
  ```

  con lo que le indicamos al ordenador que reserve en su memoria espacio para 6 variables con prefijo común **L$**, pudiendo tener cualquiera de ellas un máximo de 13 caracteres.

- Cuando el niño esté completamente familiarizado con la instrucción **DIM** y la emplee con soltura, puede ampliar la versatilidad de sus programas combinando esta instrucción con otras, por ejemplo **INPUT.**

DIM

Así, unas líneas de programa para introducir a través del teclado el número de elementos de la lista serían:

```
10  INPUT "CUANTOS LIBROS VAS A
    LISTAR?"; A
20  DIM L$ (A)
30  FOR N = 1 TO A
40  INPUT L$ (N)
50  NEXT N
```

- La instruccion **DIM** se utiliza también frecuentemente para la realización de tablas. Aunque su explicación escapa a los objetivos de este libro, se expone a continuación un breve ejemplo. El BASIC permite la utilización de variables con dos o más subíndices.

  Así, el programa

```
10  DIM A (2, 4)
20  FOR X = 1 TO 2
30  FOR Y = 1 TO 4
40  PRINT "VALOR"; X; ","; Y; "DE LA
    MATRIZ A"; "  ";
50  INPUT A (X, Y)
60  PRINT A (X, Y)
70  NEXT Y
80  NEXT X
```

  definiría los valores de una tabla similar a esta:

DIM

	Columna 1	*Columna 2*	*Columna 3*	*Columna 4*
Fila 1	A (1, 1)	A (1, 2)	A (1, 3)	A (1, 4)
Fila 2	A (2, 1)	A (2, 2)	A (2, 3)	A (2, 4)

De esta forma, el elemento A (2, 3) será el situado en la intersección entre la fila 2 y la columna 3.

Ordenando cosas

Una de las ventajas que tienen los ordenadores es que pueden ayudarte mucho a hacer trabajos en los que hay que repetir una misma cosa muchas veces. Como ya sabes, el ordenador es muy rápido. Así que, un trabajo sencillo que a tí te puede llevar mucho tiempo, el ordenador lo hace en segundos.

Nosotros a lo largo del día siempre estamos ordenando algo: Ordenamos las pegatinas, ordenamos las fichas de clase, etc. También podríamos ordenar los nombres de nuestros amigos por orden alfabético en nuestro listín. Y tu ordenador puede también ordenar todas estas cosas por tí. Sólo tienes que enseñarle cómo hacerlo.

★ Vamos a ir viendo poco a poco cómo se hace. Entrenemos primero al ordenador para que vaya comparando números. Escribe este programa para que el ordenador compare dos números dados.

Ordenando cosas

```
10  CLS
20  INPUT "DI UN NUMERO"; K
30  INPUT "DI OTRO NUMERO"; A
40  IF A > K THEN LET K = A
50  PRINT "EL MAYOR ES"; K
```

EN LA LÍNEA 40 K TOMA EL NÚMERO MAYOR, QUE ES EL QUE VAMOS A IMPRIMIR

En las líneas 20 y 30 el ordenador espera los valores. En la línea 40 el ordenador los compara. Le decimos que si el valor que hemos dado a la variable A es mayor que el valor de la variable K, entonces que la variable K tome el valor de A (que es mayor) y lo imprima.

Por ejemplo K = 5 y A = 9.

Como A es mayor que K (es decir, 9 es mayor que 5) K = A (K toma el valor 9) y se imprime.

Pueba tú cambiando los valores de A y K.

Si queremos que el ordenador compare varios números y después nos escriba el mayor, podemos utilizar un bucle. Añade las líneas:

```
24  INPUT "CON CUANTOS NUMEROS LO QUIERES
    COMPARAR?"; N
25  FOR I = 1 TO N
45  NEXT I
```

Así puedes darle hasta N números para comparar. El ordenador te escribirá el mayor.

Ordenando cosas

★ Supongamos que queremos que el ordenador compare y escriba todos los números que le demos en orden (de menor a mayor).

En el programa anterior el ordenador sólo guardaba en memoria el valor mayor de los dos que comparaba. Es como si tú tuvieras una ficha con un número en tu mano. Te van dando otras fichas, y tú las comparas y te guardas solamente la ficha mayor. La menor, la tiras.

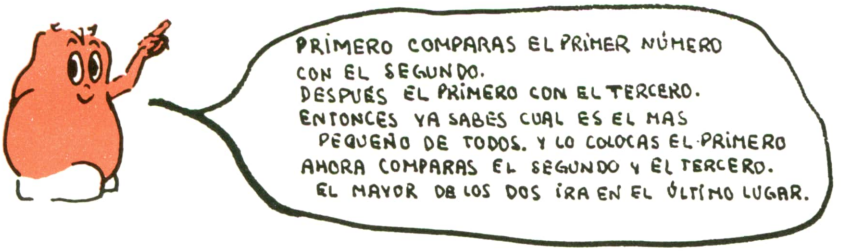
Pero si lo que quieres es ordenarlas desde el número menor al número mayor, tendrás que guardártelas todas para luego clasificarlas y ordenarlas. Así que no puedes tirar ninguna ficha o te faltarán números.

Lo mismo tiene que hacer el ordenador. Vamos a ver un ejemplo. Supongamos que queremos ordenar de menor a mayor los números que nos dan en este orden: 9, 3 y 5.

Tú con tus fichas harías así: Compararías la primera con la segunda y las ordenarías, después tomarías la que ahora ocupa el primer lugar, la compararías con la tercera y las ordenarías. Finalmente compararías las que ahora ocupen los lugares segundo y tercero y las colocarías en su lugar.

Ordenando cosas

PASOS	ORDEN
Al empezar, tenemos	9 - 3 - 5
1° Compara 9 y 3. Como 3 es menor, lo colocamos delante.	3 - 9 - 5
2° Compara 3 y 5. Como 3 es menor, lo deja donde está.	3 - 9 - 5
3° Compara 9 y 5. Como 5 es menor, lo colocamos delante.	3 - 5 - 9



La ordenación está hecha. El proceso es el mismo para 3 ó 1000 números, o los que quieras.

Pero veamos como lo haría el ordenador. Antes tendremos que asignar estos valores a unas variables.

A	B	C
9	3	5

Ordenando cosas

Cuando trabajemos con el ordenador, para obtener el resultado final tendremos que darle una orden del tipo **PRINT** A, B, C. Pero ¡OJO! Si el ordenador escribiera ahora los valores de estas variables, escribiría 9 - 3 - 5. Así que tenemos que decirle que asigne a la variable A el número más pequeño, a la variable B el número intermedio y a la varible C el número mayor.

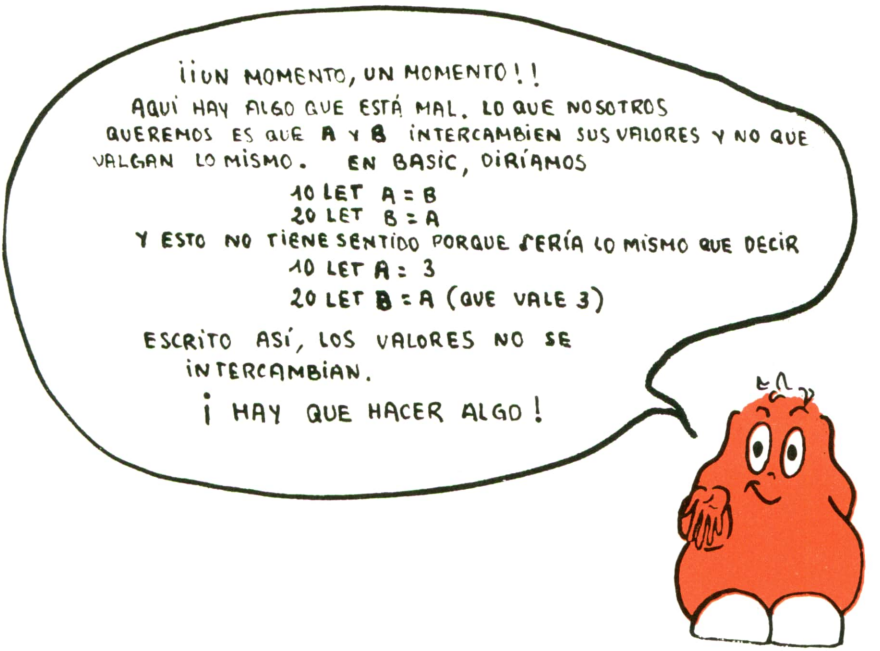
Es decir, que el ordenador tendrá que intercambiar los valores de las variables entre sí. Los pasos a seguir serían los siguientes:

A	B	C
9	3	5

1° compara A y B. Como 3 es menor que 9, lo asignamos a A. Es decir, A = 3, que es lo mismo que decir que A toma el valor de B (A = B).

A	B	C
3	3	5

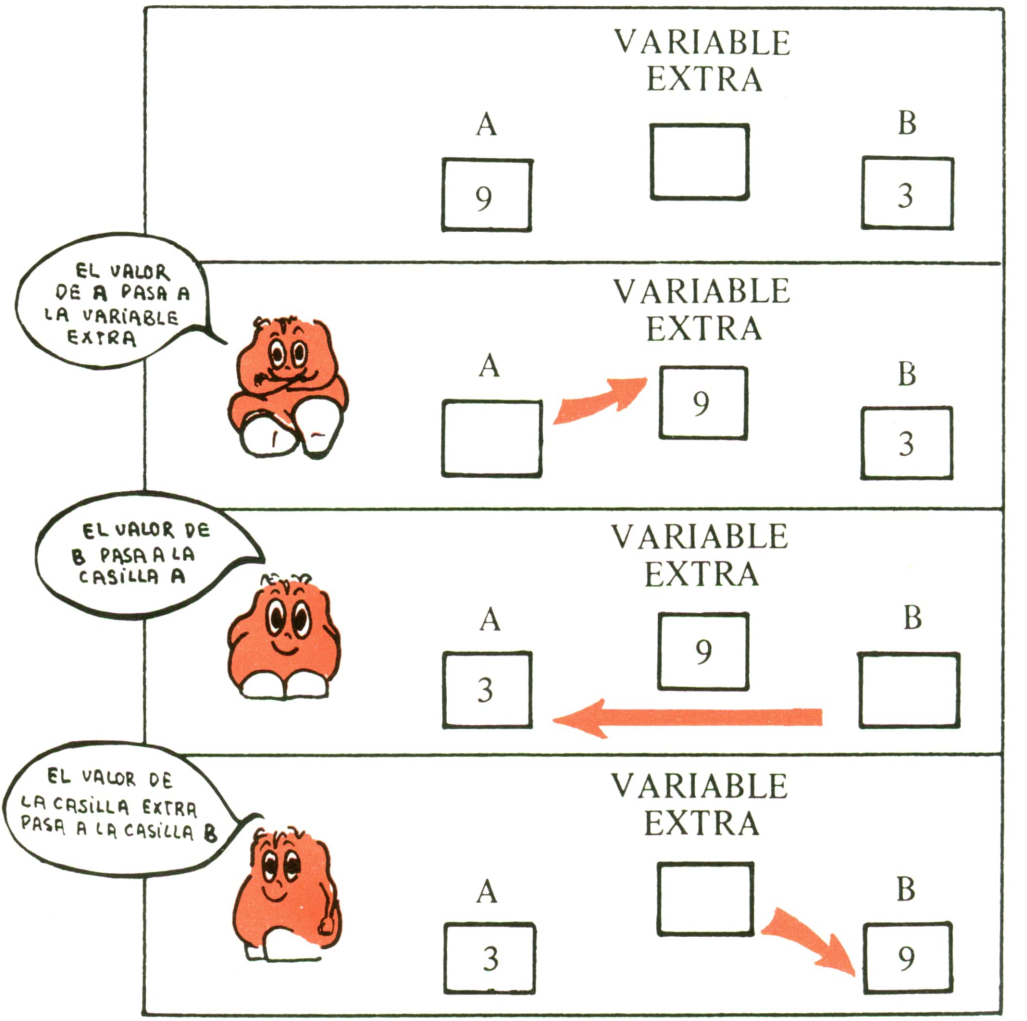
Ordenando cosas



Tiene razón ARTURO. Tenemos que decirle al ordenador que la variable A no pierda su valor cuando tome el valor de B. Hay una forma de hacerlo: Utilizar una variable *extra* donde momentáneamente se pueda guardar ese valor.

El proceso que seguiría el ordenador sería:

Ordenando cosas



Ordenando cosas

Escrito en BASIC, el anterior proceso sería así:

```
10  LET A=9
20  LET B=3
30  PRINT A; B
40  LET E=A
50  LET A=B
60  LET B=E
70  PRINT A; B
```

De esta forma se intercambian los valores de A y B sin perder ninguno de ellos.

Puedes comprobarlo fácilmente. Si ejecutas el programa verás como el ordenador imprime:

```
9    3
3    9
```

Con las líneas 40, 50 y 60 has intercambiado los valores de las variables A y B. Este truco es muy importante pues es la clave para poder desarrollar todos los programas que te explicamos a continuación. También podías haber escrito estas líneas utilizando una sola línea:

```
40  LET E=A: LET A=B : LET B=E
```

Ordenando cosas

Sigamos pues analizando el proceso que interrumpimos en la página 217 cuando ARTURO nos avisó que lo estábamos haciendo mal. Vamos a empezar otra vez pero utilizando una casilla extra (E). El ordenador actúa así:

A	B	C	E
9	3	5	

1er PASO

Toma A (que vale 9) y lo compara con B (que vale 3). Como B es menor tiene que intercambiarlo con A. Siguiendo el proceso que vimos antes, pasa el valor de A (que vale 9) a la casilla extra E.

A	B	C	E
	3	5	9

Ahora coge el valor de B (que vale 3) y lo pasa a la casilla A (que ahora está vacía).

A	B	C	E
3		5	9

Ordenando cosas

Luego pasa el valor de la casilla extra (que ahora es 9) a la casilla B.

A	B	C	E
3	9	5	

Al terminar este paso el ordenador ha comparado el primero y segundo elementos y ha colocado en la variable A el menor de ellos.

2° PASO

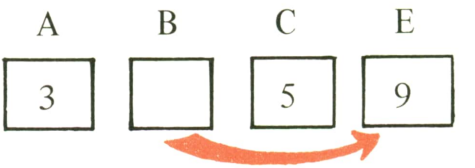
Toma A (que ahora es 3) y lo compara con C (que vale 5). Como A es menor que C, no tiene que intercambiarlos. Ya hemos fijado que el elemento menor es 3, pues ya lo hemos comparado con los otros dos. La variable A tiene ahora el menor valor.

A	B	C	E
3	9	5	

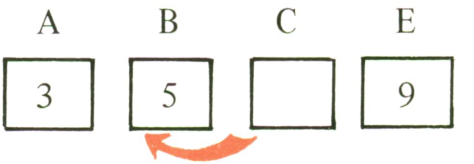
Ordenando cosas

3er PASO

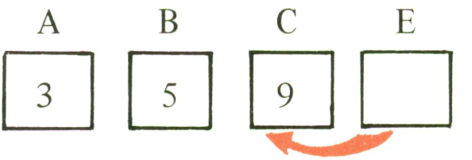
Toma B (que ahora es 9) y lo compara con C (que vale 5). Como C es menor tiene que intercambiarlo con B. Pasa el valor de B (que vale 9) a la casilla extra.



Ahora coge el valor C (que es 5) y lo pasa a la casilla B (que ahora está vacía).



Luego pasa el valor de la casilla extra (que ahora es 9) a la casilla C.



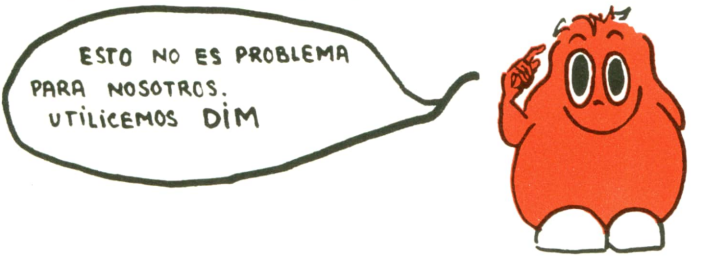
Ya hemos colocado los tres números de menor a mayor.

Ordenando cosas

Todo este proceso escrito en forma de programa, podría ser parecido a éste:

```
10  INPUT A, B, C
20  IF A>B THEN LET E=A: LET A=B: LET B=E
30  PRINT A ;" ";B;" ";C
40  IF A>C THEN LET E=A : LET A=C : LET C=E
50  PRINT A;" ";B;" ";C
60  IF B>C THEN LET E=B: LET B=C: LET C=E
70  PRINT A: " ";B;" ";C
80  END
```

Estarás pensando que el programa resulta un poquito largo para ordenar solamente 3 números. Si tuviéramos que ordenar 100 números, nos resultaría más cómodo ordenarlos a mano. Así que tendremos que pensar algo...



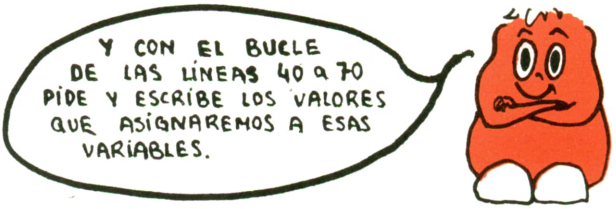
★ ¡Es cierto! Con **DIM** podemos hacerlo mucho más fácil. Tú ya sabes cómo utilizarlo, así que vamos a analizar el programa poco a poco.

Ordenando cosas

Primero escribe:

```
10  CLS
20  INPUT "CUANTOS NUMEROS QUIERES ORDE-
    NAR?"; N
30  DIM A (N)
40  FOR I = 1 TO N
50  INPUT "DIME UN NUMERO Y PULSA ENTER"; A (I)
60  PRINT A (I); " ";
70  NEXT I
75  PRINT : PRINT
```

Te recuerdo que con la instrucción **DIM** de la línea 30 preparas al ordenador para que reserve en su memoria N casillas, que es el número de variables que vamos a utilizar.

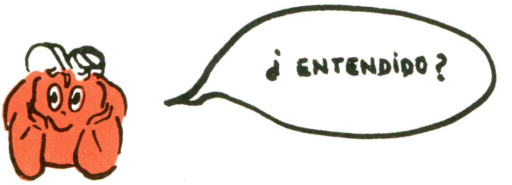


Por ejemplo, corre el programa y dale a N el valor 5. El ordenador va a comparar 5 números. Con el bucle iniciado en la línea 40 el ordenador recibirá los 5 valores que tú metas a través del teclado y los irá asignando a las variables. Por ejemplo, introduce esta serie de números: 5 - 9 - 2 - 3 - 1.

Ordenando cosas

El ordenador los asignará de esta forma:

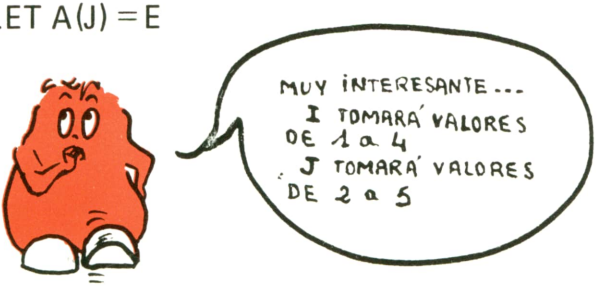
```
A (1) = 5
A (2) = 9
A (3) = 2
A (4) = 3
A (5) = 1
```



El ordenador te los escribirá en pantalla.

Seguimos. Presta mucha atención, porque ahora vienen unas líneas que son muy bonitas:

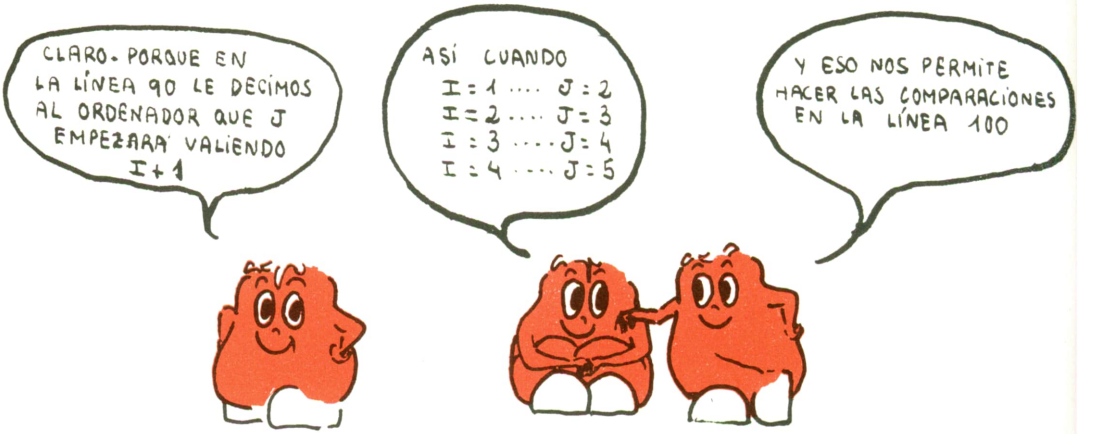
```
 80   FOR I = 1 TO N - 1
 90   FOR J = I + 1 TO N
100   IF A (I) > A (J) THEN LET E = A (I): LET A (I) =
      = A (J): LET A(J) = E
110   NEXT J
120   NEXT I
```



Ya conoces cómo funciona la línea 100 porque es donde el ordenador compara e intercambia valores ayudándose de la casilla extra.

Ordenando cosas

Fíjate que los valores que compara son A (I) y A (J). Con las líneas 80 y 90 iniciamos unos bucles que darán respectivamente valores a A (I) y A (J). Estos bucles están muy bien pensados. Verás. El bucle que empieza en la línea 90 está anidado dentro del bucle que empieza en la línea 80. Así que mientras I vale 1, J irá tomando sucesivamente los valores 2, 3, 4 y 5.



Fíjate otra vez en la línea 100. Primero compara A (1) con A (2) o sea, 5 y 9, e intercambiará sus valores si es necesario. Sigue a la línea 110 y toma el siguiente valor de J. Ahora compara A (1) con A (3), y así sucesivamente...

Cuando J ya vale 5, llega a la línea 120 y toma el siguiente valor de I (que es 2). Ahora comparará A (2) con A (3) y así sucesivamente...

Ordenando cosas

¡Ojo! Cuando el ordenador compara A (1), A (2), etc... lo que compara en realidad son los valores que A (1), A (2), etc... tienen en ese momento.

Utilizamos I y J únicamente para marcar el subíndice y así poder comparar el primer elemento de la serie con el segundo y siguientes y luego el segundo con los siguientes, etc.

Solamente nos falta añadir unas líneas para que nos imprima los resultados. Escribe:

```
130  FOR I=1 TO N
140  PRINT A (I);" ";
150  NEXT I
160  END
```

Ordenando cosas

NOTAS PARA EL ADULTO

- Los conceptos vertidos en este capítulo alcanzan ya un cierto grado de complejidad. Este capítulo no debe abordarse si no se dominan completamente las técnicas de utilización de la instrucción **DIM** explicadas en el capítulo anterior.

- Si lo considera oportuno, dedíquele más de un día o numerosas sesiones a este tema, asegurándose de que el niño va asimilando paulatinamente las distintas fases del proceso de comparación y ordenación de elementos.

- Es importante que vea la diferencia entre el proceso normal de ordenación y el proceso del computador, que requiere la utilización de una casilla o variable extra. Asegúrese de que el niño ha captado la forma de utilización de la variable extra.

- Haga ver al niño que el ordenador siempre comparará el primer elemento de una serie con todos y cada uno de los restantes, para fijar la posición definitiva de ese elemento (en nuestro caso, el menor). Una vez fijado el menor de todos, comenzará de nuevo el proceso con el siguiente elemento de la serie.

Ordenando cosas

- La clave del programa radica en la estructura de las líneas 80 a 120. Dediquen todo el tiempo que sea necesario a la comprensión de los dos bucles. Repasen si es preciso los capítulos anteriores sobre **FOR/NEXT** y **DIM.**

- Recalque que las variables I y J de dichos bucles sirven únicamente para definir A (1), A (2), A (3)... y para que el ordenador pueda comparar sus valores. No obstante, es muy importante significar que A (1), A (2), etc... tienen cada una su valor, que ha sido asignado a través del teclado al ejecutar el programa.

- En nuestro ejemplo hemos ordenado los números de menor a mayor. Para ordenar de mayor a menor, basta con cambiar el operador de relación > por el operador de relación < en la línea 100.

INT (Algo más)

Recordarás que con **INT** obteníamos la parte entera de un número. Vamos a recordarlo con un programa:

```
10  FOR A=10 TO 100 STEP 10
20  LET B=A/2.34
30  PRINT B
40  NEXT A
```

Este programa toma números de diez en diez, los divide entre 2,34 e imprime el resultado. Como podrás ver al correr el programa, los resultados son números decimales.

Si utilizamos **INT**, podemos quitar la parte decimal. Añade la línea:

```
25  LET C=INT (B)
```

Y cambia la línea 30 así:

```
30  PRINT B;" ";C
```

INT (Algo más)

Ejecuta el programa. Te queda solamente la parte entera de los resultados. Sin embargo, muchas veces es mejor redondear hacia arriba o hacia abajo en lugar de quitar la parte entera.

★ ¡Claro! Suponte que los datos que estemos utilizando sean importantes en algún cálculo. El número 29,9145299 es casi 30, pero con **INT** lo dejamos en 29.

Es muy fácil. Solamente tenemos que sumar 0,5 al número antes de quitarle la parte decimal. Por ejemplo:

42.7350427 0.5	Como el número está cerca de 43, al sumarle 0.5
43.2350427	se pasa de 43
INT (43.2350427)	y con INT queda sólo 43

34.1880342 0.5	Como el número está cerca de 34 al sumarle 0.5
34.6880342	no llega a 35
INT (34.6880342)	y con INT se queda en 34.

INT (Algo más)

Modificando el programa de acuerdo a esta regla, añadiríamos:

```
27  LET R = INT (B + 0.5)
 5  PRINT "DECIMAL"; "  "; "ENTERO"; " "; "RE-
    DONDEADO"
```

y cambiaríamos la línea 30 por:

```
30  PRINT B; "  "; C; "  "; R
```

Al correr el programa tendrás una pantalla con tres columnas:

Número Decimal	Parte Entera	Número Redondeado
4.27350427	4	4
8.54700855	8	9
.....	....	.....
.....	....	.....

Observa las diferencias entre una columna y otra.

INT (Algo más)

NOTAS PARA EL ADULTO

Este capítulo no se ha limitado a comentar la función **INT** (ya conocida) sino una forma de utilizarla. Mucho más importante que su dominio o eventual aplicación posterior es el hecho de que el niño comprenda que utilizando nuestra imaginación hemos podido transformar la forma de trabajar de una función conocida en otra acorde a nuestros intereses.

En efecto, la fórmula general **LET** A = **INT** (X + 0.5) es equivalente a una supuesta nueva función en BASIC que automáticamente redondearía cualquier número.

De esta forma, podemos continuar combinando determinadas funciones del BASIC con expresiones matemáticas sencillas para buscar nuevas aplicaciones o soluciones a problemas concretos que pudieran surgir en el transcurso de nuestros programas.

Así, si tuviéramos que determinar dentro de un programa cuándo un número entero es par o impar, utilizaríamos la fórmula:

```
10  IF INT (X/2) * 2 = X THEN PRINT " NUME-
    RO PAR"
```

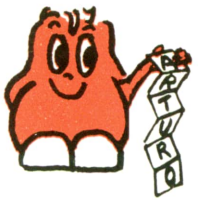
la cuál se explica por sí sola.

INT (Algo más)

- Aparte de **INT** y **RND** existen otras funciones numéricas que no se mencionan pues rara vez serán utilizadas por el niño a este nivel. Resumimos a continuación la aplicación de dichas funciones:

SGN (N)	Indica el signo de N	SGN (3) hará aparecer en pantalla 1 SGN (–2) hará aparecer en pantalla –1 SGN (0) hará aparecer en pantalla 0
ABS (X)	Indica el valor absoluto de un número	ABS (–7) hará aparecer en pantalla 7 ABS (7) hará aparecer en pantalla 7
SQR (N)	Halla la raiz cuadrada de N	SQR (9) hará aparecer en pantalla 3

- El BASIC ofrece también diversas funciones logarítmicas, exponenciales y trigonométricas que facilitan considerablemente trabajos de cálculo. No se citarán en este libro.



Cadenas...

Aparte de las instrucciones (**PRINT**, **IF**, **GOTO**, **LET**, **FOR/NEXT**, **READ** ...) hasta ahora hemos conocido también funciones numéricas como **INT** o **RND**.

Vamos a conocer ahora algunas funciones de cadena o funciones que trabajan con textos. Son muy útiles, pues con ellas el BASIC te permite trabajar con palabras, frases, etc.

No las vamos a explicar todas. Solamente las que ahora te pueden resultar más interesantes. Además, no profundizaremos mucho en ellas. Lo principal es que sepas cómo funcionan y para qué pueden servir.

LEN nos dice el número de caracteres que tiene una cadena. Por ejemplo:

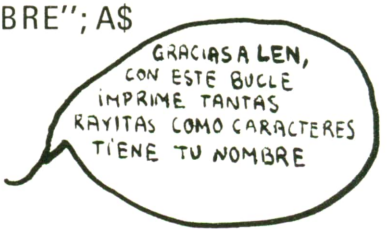
```
10  LET A$ = "BASIC PARA NIÑOS"
20  PRINT LEN (A$)
```

Cadenas...

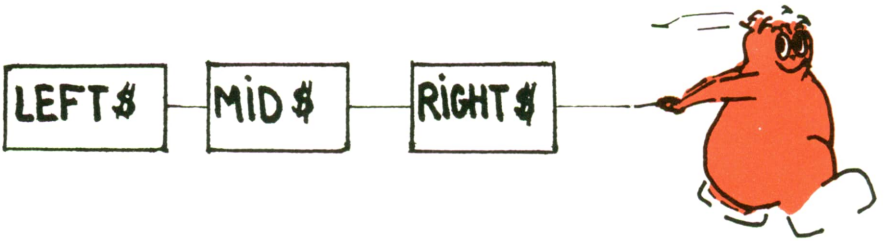
Al correr el programa, en tu pantalla aparecerá el número 16. **LEN** ha contado todos los espacios de la cadena alfanumérica A$, incluidos los espacios en blanco.

LEN puede ser muy útil cuando quieras subrayar una palabra que previamente no conoces. Por ejemplo, teclea:

```
10  INPUT "ESCRIBE TU NOMBRE"; A$
20  PRINT A$
30  FOR N=1 TO LEN (A$)
40  PRINT " – ";
50  NEXT N
```

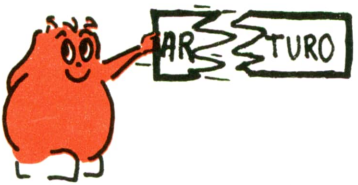


Toda la cadena aparecerá subrayada.



Estas funciones nos permiten obtener unos caracteres concretos sacándolos de una cadena. Por ejemplo **LEFT$** extrae los caracteres que tú le indiques, empezando a contar por la izquierda de la cadena. **RIGHT$** hace lo mismo, pero empezando por la derecha. Veamos un ejemplo: Teclea:

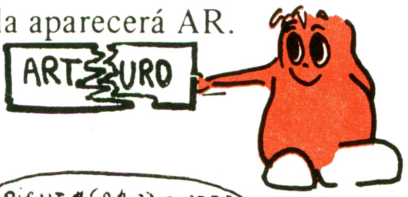
Cadenas...



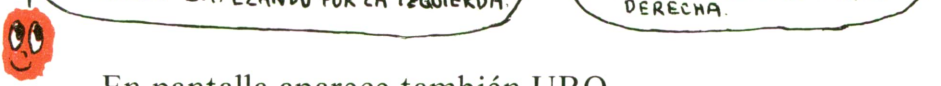

```
10  LET A$ = "ARTURO"
20  PRINT LEFT$ (A$, 2)
```

Corre el programa y en la pantalla aparecerá AR.

Añade la línea 30:




```
30  PRINT RIGHT$ (A$, 3)
```



En pantalla aparece también URO.

Hagamos un ejemplo de aplicación más útil:

```
10  INPUT "  ESCRIBE UN VERBO EN INFINITIVO.
    YO TE DIRE A QUE CONJUGACION PERTENECE";
    A$
20  LET B$ = RIGHT$ (A$, 2)
30  IF B$ = "AR" THEN PRINT A$; " "; "ES DE LA
    PRIMERA"
40  IF B$ = "ER" THEN PRINT A$; " "; "ES DE LA
    SEGUNDA"
50  IF B$ = "IR" THEN PRINT A$; " "; "ES DE LA
    TERCERA"
```

Cadenas...

Fíjate que en la línea 20 asignamos a la variable B$ el valor de los dos últimos caracteres de la variable A$. En informática se dice que B$ es una *subcadena* de A$.

MID$ es una función más perfeccionada que las anteriores. Fíjate en el ejemplo:

```
10  LET A$="ARTURO"
20  PRINT MID$ (A$, 2, 3)
```

Al ejecutar el programa en tu ordenador aparecerá RTU.

Ya conoces como funcionan **LEN**, **LEFT$**, **RIGHT$**, y **MID$**. Vamos a hacer un programa fenomenal combinando dos de estas funciones. Teclea:

```
10  INPUT "ESCRIBE UN TEXTO LARGO"; A$
20  CLS
30  LET L=LEN (A$)
40  FOR N=1 TO L
50  PRINT MID$ (A$, N, 1);
60  FOR X=1 TO 50: NEXT X
70  NEXT N
```

Cadenas...

En la línea 30 la variable L va a tomar como valor el número de caracteres de tu texto, incluidos los espacios en blanco.

En la línea 40 se inicia un bucle en el que N valdrá desde 1 hasta el número total de caracteres de tu texto.

En la línea 50 el ordenador toma de uno en uno los sucesivos caracteres de tu texto, empezando por el primero (puesto que N empieza valiendo 1) y los imprime uno al lado del otro.

El bucle de retardo de la línea 60 marcará la velocidad con que aparecerá tu texto en la pantalla.

Esta función nos permite asignar un valor a una variable alfanumérica a través de nuestro teclado sin necesidad de pulsar **ENTER** o **RETURN** después de hacerlo.

Por ejemplo, escribe:

```
 5  PRINT "PULSA CUALQUIER TECLA"
10  LET A$=INKEY$: IF A$=" " THEN 10
```

Con esta línea el ordenador espera un valor que tú darás a través del teclado y que será asignado a la variable A$.

Cadenas...

En la segunda sentencia de la línea 10 le decimos al ordenador que mientras A$ no valga nada (es decir, mientras no pulses una tecla del teclado) continúe esperando en esa misma línea. Sigue:

```
20  PRINT " PULSASTE LA TECLA ...";A$
30  GOTO 5
```

Corre el programa y prueba pulsando varias teclas. Verás que no es necesario pulsar **ENTER** para que el dato quede introducido en el programa.

El truco de la línea 10 es además muy práctico para parar un programa mientras se está ejecutando y que no continúe hasta que pulsemos una tecla.

★ ¿Recuerdas el programa del AJEDREZ de la página 143?

Con las líneas 10, 90 y 100 hacíamos que el ordenador imprimiera en pantalla los granos de cada nueva casilla siempre que tú teclearas S y después **ENTER** o **RETURN**.

Con **INKEY$** podemos hacer el programa más cómodo. Cambia las líneas 10 y 80 por:

```
10  PRINT " PULSA CUALQUIER TECLA PARA
        OTRA CASILLA"
80  LET A$=INKEY$: IF A$=" " THEN 80
```

Cadenas...

Quita las línea 90 y 100 del programa antiguo. El resultado es el mismo y la ejecución del programa más cómoda.

Puedes hacer lo mismo con el programa de la página 171, que nos daba la tabla de multiplicar (cambia la línea 90).

NOTAS PARA EL ADULTO

Siga los ejemplos con el niño. En principio, éste no debe encontar ninguna dificultad especial en la comprensión de los programas expuestos y en el empleo de las funciones mencionadas.

Algunos ordenadores (SINCLAIR) utilizan una expresión simplificada de las funciones **LEFT$, RIGHT$** y **MID$**. Así, considerando A$ = "ARTURO":

PRINT A$ (2 TO 4) escribirá RTUR
PRINT A$ (3) escribirá T
PRINT A$ (TO 4) escribirá ARTU
PRINT A$ (3 TO) escribirá TURO

Así, la línea 20 del programa de la página 237 se escribirá:

20 LET B$ = A$ (LEN A$ – 1 TO)

Cadenas...

la cual asigna igualmente a B$ los dos últimos caracteres de la variable A$.

Igualmente, la línea 50 del programa de la página 238 se escribirá:

```
50 PRINT A$ (N);
```

Tenga en cuenta también que deberá variar la colocación de los paréntesis en función del ordenador que utilice.

- Existen otras funciones de cadena, –cuya explicación escapa a los objetivos de este libro– como serían:

VAL (A$) Convierte la cadena numérica A$ en su valor numérico.

Ejemplo:

```
10  LET A$ = "234"
20  LET A = VAL (A$)
30  PRINT A
```

En la pantalla aparecerá 234, pero no como cadena, sino como número,

STR$ (A$) Convierte un número en una cadena de caracteres numéricos. Permite así aprovechar en un número las ventajas del tratamiento de textos a través de las funciones de cadena.

Ejemplo:

```
10  LET A =234
20  LET A$ =STR$ (A)
30  PRINT A$
```

En la pantalla aparecerá 234, pero no como número, sino como cadena. Prueba de ello es que añadiendo la línea:

```
40  PRINT RIGHT$ (A$, 2)
```

El ordenador escribirá:

```
234
34
```

- En algunos ordenadores, la función **INKEY$** se suple con la función equivalente **GET.** Aunque en sí, esta función no es una función de cadena, se emplea fundamentalmente en el tratamiento de cadenas. Así, **GET** permite introducir un carácter numérico y **GET$** un carácter alfanumérico. Obviamente, se suele utilizar más frecuentemente esta última, ya que de este modo el ordenador acepta cualquier tecla que se pulse.

  Esta función adopta la forma:

```
10  GET A$
```

CHR$ y ASC

Todos los números, letras, sígnos, etc, que aparecen en el teclado de tu ordenador se corresponden con unos números que tiene el ordenador memorizados.

La relación entre números y caracteres se llama *código.* Este código puede cambiar de un ordenador a otro.

Las funciones **CHR$** y **ASC** nos permiten ver la relación que existe entre esos números y los símbolos o caracteres del teclado.

Por ejemplo PRINT ASC ("A") nos dará el número de código que tiene el carácter "A" en tu ordenador.

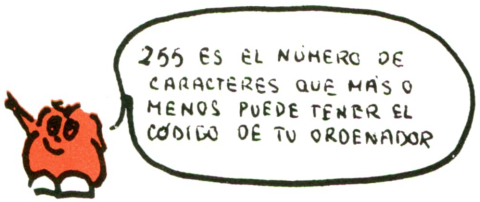
PRINT CHR$ (65) nos dará el carácter que corresponde al número 65 en el código de tu ordenador.

CHR$ y ASC

A veces es conveniente conocer la relación entre el carácter y el número de código. Vamos a ver un ejemplo, haciendo un programa para jugar.

★ Antes que nada, vamos a ver cuál es el código de tu ordenador. Teclea:

```
10  FOR N=1 TO 255
20  PRINT CHR$ (N), N
30  NEXT N
```

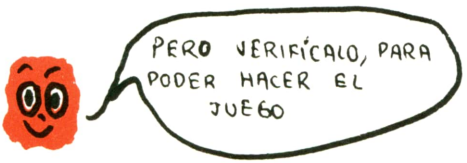


Si no te da tiempo a verlos todos, añade un bucle de retardo:

```
25  FOR I=1 TO 200: NEXT I
```

Corre el programa. En la columna de la izquierda te aparecerán los símbolos, números, caracteres y letras de tu teclado. En la columna de la derecha, sus correspondientes números de código.

Fíjate, por ejemplo, en el número de código que tienen las letras del abecedario. Posiblemente en tu ordenador las letras del abecedario se correspondan con los números 65 hasta el 90.



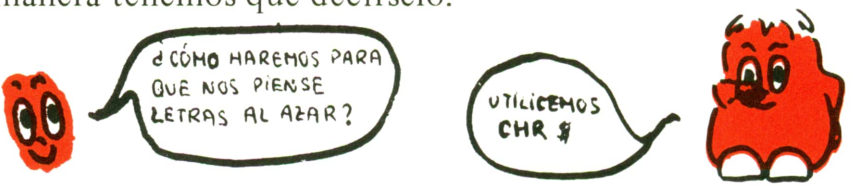
CHR$ y ASC

★ Ahora vamos con el juego. Primero lo haremos con las instrucciones que ya conocemos. Teclea:

```
10  LET A = INT ( RND (X) * 10)
20  INPUT " HE PENSADO UN NUMERO DEL 1 AL 10.
    ADIVINALO. ESCRIBE TU NUMERO"; Z
30  IF A <> Z THEN 20
40  PRINT "ACERTASTE"
```

Este es un juego muy sencillo que sin duda ya entiendes.

★ Pero suponte que en lugar de jugar a adivinar números queremos que el ordenador piense en una letra. De alguna manera tenemos que decírselo.



¡Claro! **CHR$** transformará los números de código que le demos en sus correspondientes caracteres. Como sabemos que, en nuestro ejemplo, las letras del alfabeto van dentro del código desde el número 65 al 90, escribiremos:

```
10  LET A = INT (RND (X) * 26) + 65
15  LET A$ = CHR$ (A)
```

CHR$ y ASC

Fíjate en estas líneas. En la línea 10, hacemos que **RND** (X) * 26 genere números al azar entre 1 y 25 (que es el número de letras que tiene el alfabeto).

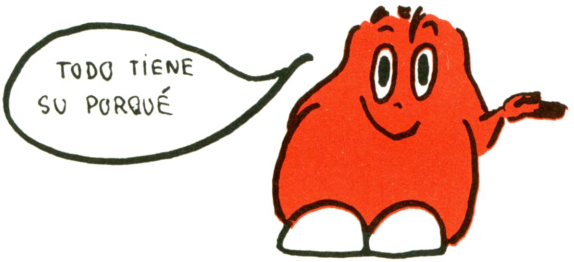
Después a ese número le sumamos 65 porque la letra A se corresponde con el número 65 del código. Así, al sumar 65 al número obtenido con **RND**, obtendremos al azar números comprendidos entre 65 y 90.

Con **CHR$** hacemos que el ordenador transforme dicho número de código a su letra correspondiente.

Añade:

```
20 INPUT "HE PENSADO UNA LETRA DEL ABECE-
   DARIO. INTENTA ADIVINARLA. ESCRIBE TU
   LETRA"; Z$
30 IF A$<>Z$ THEN 20
40 PRINT "ACERTASTE"
```

Al correr el programa, el ordenador tomará al azar una letra cualquiera del abecedario. Como ves, **CHR$** puede ser útil.



CHR$ y ASC

Puedes completar el programa añadiendo contadores que te digan cuánto tardas en adivinarlo, dando puntuaciones, etc. Esto ya sabes como hacerlo.

NOTAS PARA EL ADULTO

- Estas instrucciones no serán frecuentemente utilizadas por el niño, pero las incluímos a fin de que, al menos, le resulten familiares. Al mismo tiempo se ha efectuado una introducción al código ASCII, que es un código internacional utilizado para representar informaciones alfanuméricas.

- El código ASCII está estandarizado y las informaciones alfanuméricas fundamentales suelen tener siempre el mismo código en todas las computadoras. No obstante, algunos fabricantes introducen ligeras variaciones por,lo que le sugerimos que antes de abordar los programas de este capítulo consulte el manual de instrucciones de su computador, corra el programa de la página 245 o teclee **PRINT ASC** ("A") y **PRINT ASC** ("Z") para determinar el código del alfabeto de su ordenador.

CHR$ y ASC

No olvide que no todos los ordenadores expresan de igual forma la instrucción **RND.** Posiblemente su ordenador precise escribir la línea 10 así:

```
10    LET A = RND (25) + 65
```

o bien:

```
10  LET A = INT (RND * 25) + 65
```

Algunos ordenadores utilizan la función **CODE** en lugar de **ASC.** En cualquier caso no olvide que esta función nos indicará el número de código correspondiente al primer carácter de la expresión alfanumérica.

Así:

```
ASC ("ANTONIO") = 65
ASC ("A") = 65
```

Por otra parte, algunos ordenadores escribirán:

```
ASC "ANTONIO" en lugar de ASC ("ANTONIO")
CHR$ 65       en lugar de CHR$ (65)
```

Para programar bien

Después de leer este libro tú sabes ya muchas cosas sobre el BASIC. Serás capaz de hacer programas bastante amplios. Te gustará que tus programas sean buenos, funcionen, sean cómodos de utilizar, estén escritos con lógica...

Cuando ARTURO empezó a programar, tuvo algunas dificultades. No todos sus programas funcionaban a la primera. En la pantalla le aparecían avisos de error, etc.

En realidad, sólo con la práctica se consigue poco a poco programar con mayor seguridad. De todas formas, vamos a darte algunos consejos para que tus programas sean realmente buenos.

- Antes que nada conviene que tengas muy claro qué es lo que quieres hacer. Cuanto más domines un tema, más fácil te será hacer un programa relativo a ese tema.

Para programar bien

● Después te resultará útil hacer un diagrama de flujo. Con ello tendrás una visión general de cómo va a ser tu programa.

● Puede resultarte cómodo escribir las partes más complicadas del programa en un papel antes de pasarlo al ordenador.

● Al teclear tu programa debes tener cuidado cuando escribas los números de línea, para no repetirlos y evitar así que sin querer borres una de ellas.

● Ten mucho cuidado para no confundir las variables entre sí, y tampoco las numéricas y las alfanuméricas. Utiliza variables que te recuerden a lo que representan. Por ejemplo, L$ para Libros, R$ para Ropa, N para números, etc.

Para programar bien

- Recuerda que todas las sentencias **FOR** tienen que tener una sentencia **NEXT**.

- Vigila las instrucciones que le das al ordenador para que en un **IF ... THEN** vaya a la línea de programa adecuada.

- No te olvides de poner líneas "chivatas" para detectar los posibles errores de entrada de datos a través del teclado. Con ellas se detectan los errores justo después de haberlos cometido.

- Cuando introduzcas cambios, lista el programa para ver si el ordenador te los ha admitido correctamente, de la forma y en el lugar que querías.

- También es muy importante que tu programa quede muy claro para que pueda ser entendido por alguien que no lo haya escrito. Pon muchas sentencias **REM** para aclarar las cosas. Tú mismo las agradecerás cuando vuelvas a mirar el programa después de un tiempo.

Para programar bien

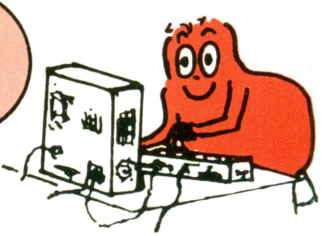
● Finalmente debes hacer que tu programa pueda ser utilizado muy fácilmente por cualquiera. Mete en el prógrama un menú que indique muy claramente las posibles opciones y los pasos a seguir.

● Tus sentencias **PRINT** deben ser muy claras. Por ejemplo, es mejor poner ¿CUANTOS CARAMELOS TIENE ARTURO? que poner ESCRIBA CARAMELOS.

Es mejor poner "EL RESULTADO DE LA MULTIPLICACION ES ..." que poner "RESULTADO ..."

En fin, tú mismo irás aprendiendo con el tiempo cómo hacer programas perfectos.

Programas

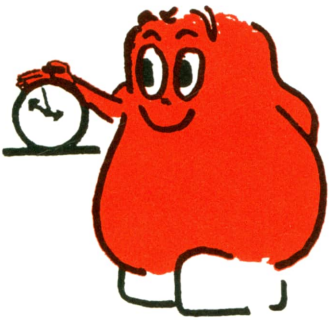


PRUEBA TUS REFLEJOS

Con este juego podrás probar tus reflejos. Intenta ganar a nuestro amigo ARTURO (que está muy entrenado). Y no te pongas nervioso...

```
 10  CLS
 20  LET Q$ = "PRUEBA TUS REFLEJOS. TE MOSTRA-
     RE UNA LETRA UN INSTANTE. EL TIEMPO EM-
     PEZARA A CORRER, Y SE PARARA CUANDO
     PULSES ESA LETRA. COMPITE CON TUS AMIGOS.
     EL QUE MENOS PUNTOS TENGA, GANA".
 30  LET L = LEN (Q$)
 40  FOR N = 1 TO L
 50  PRINT MID$ (Q$, N, 1);
 60  FOR W = 1 TO 20: NEXT W
 70  NEXT N
 80  FOR Z = 1 TO 1000: NEXT Z
 90  RANDOMIZE
100  LET X = 0
110  LET N = INT (RND (0) * 25) + 65
120  LET A$ = CHR$ (N)
130  PRINT: PRINT
140  PRINT A$
```

```
150  FOR Z=1 TO 100: NEXT Z
160  LET B$=INKEY$ : LET X=X+1: PRINT X :
     CLS : IF B$=" " THEN 160
170  IF B$=A$ THEN 200
180  IF B$<>A$ THEN PRINT " FALLASTE. INTEN-
     TALO DE NUEVO"
190  FOR Z=1 TO 200 : NEXT Z: GO TO 90
200  PRINT X : PRINT "TU PUNTUACION ES ";X
210  PRINT: PRINT: PRINT: PRINT: PRINT "¿OTRA
     VEZ? (S/N)"
220  LET R$=INKEY$ : IF R$=" " THEN 220
230  IF R$="S" THEN 90
240  IF R$="N" THEN STOP
250  IF R$<>"S" AND R$<>"N" THEN PRINT
     "CONTESTA BIEN"
260  GO TO 210
```



Programas

NOTAS PARA EL ADULTO

- De la línea 10 a la línea 70 se escribe "artísticamente" el texto de presentación en la pantalla. De la línea 90 a la línea 140 se generan aleatoriamente las letras del alfabeto, y se pone el contador a cero.

  De la línea 160 a la 180 el ordenador analiza la tecla pulsada y hace correr el marcador. De la línea 210 en adelante el ordenador permite volver a empezar.

- Conviene que añadan a este programa sentencias REM para aclarar los distintos pasos que sigue el ordenador.

- En este programa se practican de manera especial los bucles de retardo, las funciones MID$, INKEY$ y CHR$ y LEN, así como INT y RND. En algunos ordenadores, la línea 50 deberá escribirse así:

```
50   PRINT Q$ (N);
```

  y la línea 110 –que variará en función de su ordenador– se escribiría:

```
110   LET N = INT (RND * 25) + 65
```

  o también

```
110   LET N = RND (25) + 65
```

Programas

EL JUEGO DE LOS ESPIAS

Como ya has visto en las páginas anteriores, a nuestro amigo ARTURO le gusta mucho hacer de espía. Incluso ha desarrollado un programa para comunicarse en clave con su Cuartel General.

El sistema es muy sencillo. Cuando manda un mensaje, lo hace en un *código* especial. A cada letra del alfabeto le suma 5, y la letra que resulta es la que escribe. Así, en lugar de escribir una A, escribiría una F.

A	B	C	D	E	F	G	H	....
1	2	3	4	5	6	7	8	....

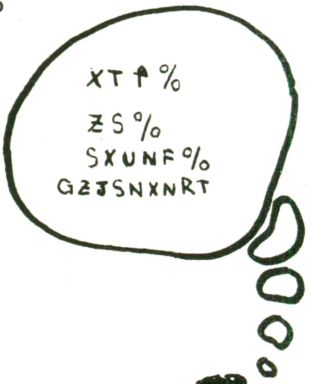
Cuando en el Cuartel General reciben un mensaje de ARTURO, lo que hacen es restarle 5 a cada letra del mensaje para poder entenderlo.

Logicamente, lo hacen con un ordenador, por ser más rápido. Tú mismo puedes establecer con tus amigos un sistema de mensajes en clave. Lo único que tienes que tener en cuenta es que hay que utilizar el mismo modelo de ordenador para luego poder descifrarlos.

Programas

El programa para codificar automáticamente los mensajes sería:

```
10  INPUT "ESCRIBE UNA FRASE"; F$
20  INPUT "INDICA LA CLAVE"; C
30  FOR X = 1 TO LEN F$
35  LET A$ = MID $ (F$, X, 1)
40  LET Z = ASC (A$) + C
50  PRINT CHR$ (Z);
60  NEXT X
70  PRINT
```



El programa para descifrarlos sería:

```
10  INPUT "ESCRIBIR FRASE CODIFICADA"; F$
20  INPUT "CLAVE SECRETA"; C
30  FOR X = 1 TO LEN F$
35  LET A$ = MID $ (F$, X, 1)
40  LET Z = ASC (A$) – C
50  PRINT CHR$ (Z);
60  NEXT X
```

En los ordenadores de SINCLAIR tendrás que cambiar la línea 35 por:

```
35  LET A$ = F$ (X TO)
```

y utilizar CODE en lugar de ASC.

Establece con tus amigos tu propio número de clave secreto y nadie podrá descubrir vuestros mensajes...

Programas

Claro que hay que tener mucho cuidado ... Si los Servicios de Contraespionaje conocen vuestro sistema de códigos, pueden averiguar incluso la clave con un programa como el que acabamos de ver, cambiando la línea 20 por

```
20  FOR C = 1 TO 24
```

y añadiendo las líneas

```
70  PRINT
80  NEXT C
90  STOP
```

Con ello el ordenador escribe en pantalla todas las posibles frases en clave. De ellas, una tendrá sentido. Intenta descubrir lo que dice ARTURO en la página anterior, utilizando este sistema de los servicios de contraespionaje.

Programas

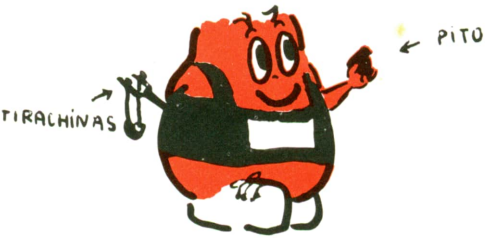
UNA LIGA DE BALONCESTO

Como te habíamos prometido antes, haremos un programa muy sencillo para organizar una liga de baloncesto. ARTURO organizó su campeonato y este programa le fue muy útil pues con él tuvo la seguridad de que no se olvidaba de incluir ningún partido. Participaban solamente tres equipos.

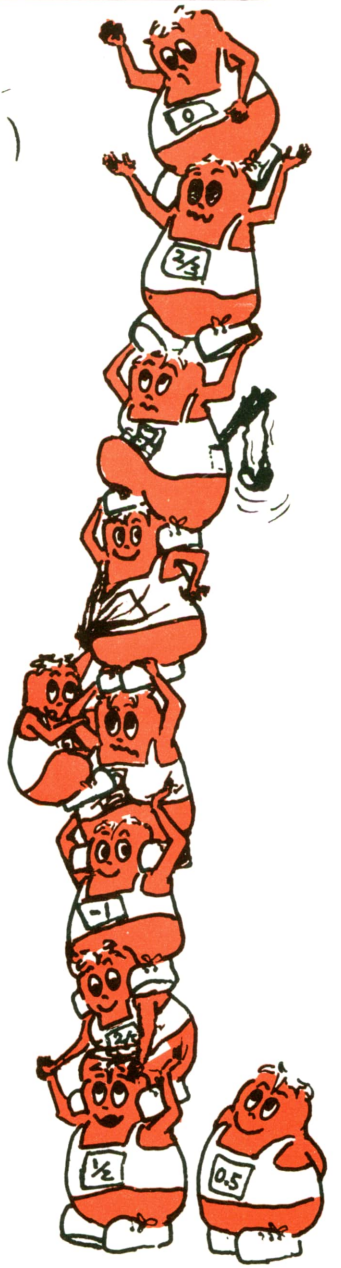
La liga de ARTURO quedó así:

PARTIDOS DE LA LIGA

ROMPETECHOS contra LARGUIRUCHOS
ROMPETECHOS contra TROTAMUNDOS
LARGUIRUCHOS contra ROMPETECHOS
LARGUIRUCHOS contra TROTAMUNDOS
TROTAMUNDOS contra ROMPETECHOS
TROTAMUNDOS contra LARGUIRUCHOS



Programas



El programa sería:

```
 10   FOR A = 1 TO 3
 20   FOR B = 1 TO 3
 30   IF B = A THEN GOTO 100
 40   LET E = A
 50   GO SUB 200
 60   PRINT " contra ";
 70   LET E = B
 80   GO SUB 200
 90   PRINT
100   NEXT B
```

```
110  NEXT A
120  STOP
200  IF E=1 THEN PRINT " ROMPETECHOS";
210  IF E=2 THEN PRINT " LARGUIRUCHOS";
220  IF E=3 THEN PRINT " TROTAMUNDOS";
230  RETURN
```

Este programa solamente te serviría para organizar una liga como la de ARTURO. Si quieres organizar tu propia liga, puedes incluir el número de equipos que desees, utilizando la instrucción **DIM**, que conocimos en la página 200.

El programa sería:

```
 5  INPUT "CUANTOS EQUIPOS"; N
10  DIM A$ (N)
20  FOR J=1 TO N
30  PRINT "ESCRIBE EL EQUIPO NO."; J
40  INPUT A$ (J)
50  CLS
60  NEXT J
70  CLS
80  PRINT "PARTIDOS DE LA LIGA"
90  PRINT
```

Programas

```
100  FOR X=1 TO N
110  FOR Y=1 TO N
120  IF X=Y THEN GOTO 150
130  PRINT A$ (X); " contra ";
140  PRINT A$ (Y)
150  NEXT Y
160  NEXT X
```

NOTAS PARA EL ADULTO

- El segundo programa permite la inclusión del número de equipos que se deseen, los cuales deberán ser introducidos a través del teclado durante la ejecución del programa.

- Algunos ordenadores (por ejemplo, SINCLAIR) exigirán una pequeña modificación en la línea 10 del programa, para definir el número máximo de caracteres que puede tener la variable alfanumérica. Así, se escribiría

```
10  DIM A$ (N, 12)
```

Programas

LISTIN TELEFONICO

Puedes hacerte tu propio listín de teléfonos y direcciones. Hay varios sistemas, pero el que te ofrecemos a continuación es muy simple:

```
   5  CLS
  10  INPUT "CUANTOS NOMBRES TIENE TU
      LISTIN?"; M
  20  PRINT "LISTIN": PRINT
  30  PRINT "BUSQUEDA POR :"
  40  PRINT "NOMBRES . . . . 1"
  50  PRINT "DOMICILIO"  . . 2"
  60  PRINT "FIN . . . . . . . . 3"
  70  INPUT "QUE OPCION DESEAS ?"; A
  80  CLS
  90  IF A=1 THEN GOTO 2000
 100  IF A=2 THEN GOTO 3000
 110  IF A=3 THEN STOP
 120  PRINT "TECLEASTE MAL TU OPCION ":
      GOTO 30
2000  INPUT "ESCRIBE EL NOMBRE"; E$
2010  RESTORE
2020  FOR X=1 TO M
2030  READ N$, C$, T$
```

```
2040  IF E$=N$ THEN GOTO 2070
2050  NEXT X
2060  IF E$<>N$ THEN PRINT " NO CONOZCO A ";
      E$: GOTO 2000
2070  PRINT N$: PRINT C$: PRINT T$
2080  INPUT "OTRO NOMBRE? (S/N) "; Z$
2090  IF Z$="S" THEN GOTO 2000
2100  INPUT " VUELTA AL MENU? (S/N) "; J$
2110  IF J$="S" THEN GOTO 30
2120  STOP
3000  INPUT " DOMICILIO"; E$
3010  RESTORE
3020  FOR X=1 TO M
3030  READ N$, C$, T$
3040  IF E$=C$ THEN GOTO 3070
3050  NEXT X
3060  IF E$<>C$ THEN PRINT "NO CONOZCO LA
      CALLE"; E$: GOTO 3000
3070  PRINT N$: PRINT C$: PRINT T$
3080  INPUT "OTRA CALLE? (S/N)"; Z$
3090  IF Z$="S" THEN GOTO 3000
3100  INPUT " VUELTA AL MENU? (S/N)"; J$
3110  IF J$="S" THEN GOTO 30
3120  STOP
```

```
5000  DATA LUIS, INDEPENDENCIA 345,  2-34-56-83
5010  DATA FERNANDO, CENTRAL 23,  2-67-98-99
5020  DATA ANA, PLAZA COLON 3,  2-55-54-42
5030  DATA ARTURO, AV. DE LA INFORMATICA 2,
      0-11-01-10
```

Los datos de tus amigos los tienes que introducir a través de sentencias DATA. Recuerda los consejos que te dábamos en la página 199 sobre las sentencias DATA. Si añades más nombres, no te olvides de cambiar la línea 10.

ORDENACION ALFABETICA

```
10  CLS
20  INPUT "CUANTOS NOMBRES QUIERES
    ORDENAR?"; N
30  DIM A$ (N)
40  FOR I = 1 TO N
50  INPUT A$ (I)
60  PRINT A$ (I)
70  NEXT I
```

```
 80  PRINT: PRINT
 90  FOR I=1 TO N-1
100  FOR J=I+1 TO N
110  IF A$ (I)>A$ (J) THEN LET E$=A$ (I): LET
     A$ (I)=A$ (J): LET A$ (J)=E$
120  NEXT J
130  NEXT I
140  FOR I=1 TO N
150  PRINT A$ (I)
160  NEXT I
170  END
```

Este programa te permite ordenar alfabéticamente los nombres de todos tus amigos. Como ves, el proceso es el mismo que para la ordenación numérica, sólo que en esta ocasión utilizamos variables alfanuméricas.

Como ya has visto en capítulos anteriores, el ordenador también clasifica las letras y puede analizar cuál es mayor. Con las palabras hace exactamente igual. Primero compara la primera letra y si ésta es igual, compara la segunda, etc...

Por ejemplo,

ANTONIO < CARLOS
ARTURO > ANTONIO

Programas

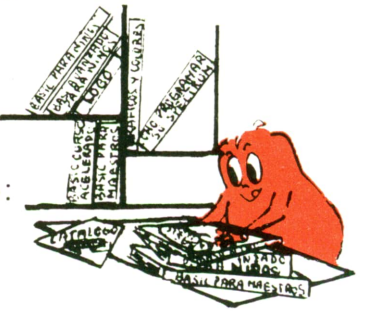
CONTROL DE BIBLIOTECA

Es muy conveniente que te acostumbres a tener tus libros ordenados y clasificados de forma que puedas localizarlos con facilidad cuando los necesites.

El ordenador te puede ayudar mucho en esto. El programa que te ofrecemos a continuación es muy eficaz y muy sencillo, aunque sea un poco largo.

Puedes utilizar hasta dos dígitos para las materias. Por ejemplo,

MEDICINA 14
LITERATURA 21



y uno para las submaterias. Por ejemplo:

NOVELAS 2

Puedes utilizar tu propia clasificación o, mejor, utilizar un sistema de clasificación que utilizan muchas bibliotecas. Pídeselo a tu maestro.

Utilizando la clave para la materia y la submateria podremos ordenar nuestros libros de manera lógica. Para clasificar una *novela* de un *autor español o hispanoamericano* utilizaríamos el código 212. También podemos poner el título y el autor. Por ejemplo:

Programas

DATA 212, JUAN RAMON JIMENEZ, PLATERO Y YO

TITULO

AUTOR

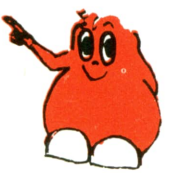
SUBMATERIA 2 (Novela española o hispanoamericana)

MATERIA 21 (Literatura)

Pon todos los datos al final de tu programa y luego puedes pedir cualquier información al ordenador: Si tecleas la materia te dará los autores y títulos que tiene tu biblioteca sobre esa materia (y además clasificados por orden alfabético). Si tecleas el autor te dará los títulos que tienes de ese autor y las materias a que corresponden. Si tecleas el título, te dirá a qué autor pertenece la obra y la materia a que corresponde.

Este programa reserva 20 caracteres para el nombre del autor y 20 caracteres para el título, por lo que cuando incluyas los datos tendrás que ajustarte a estas dimensiones.

```
 3  REM BIBLIOTECA
 4  CLS
 5  READ N
10  DIM L$ (N)
20  DIM M$ (N)
30  DIM S$ (N)
40  DIM A$ (N)
50  DIM A (N)
60  DIM T$ (N)
70  DIM T (N)
```



EN LA LÍNEA 5 LEE EL DATO DE LA LÍNEA 5000 Y PERMITE DIMENSIONAR TANTAS VARIABLES COMO NECESITEMOS (N NUMERO DE LIBROS)
LAS VARIABLES SERAN:
L$ = LIBRO COMPLETO CON TODOS LOS DATOS
M$ = MATERIA
S$ = SUBMATERIA
A$ = AUTOR
T$ = TÍTULO

Programas

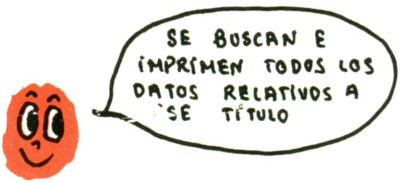
```
 80  FOR I = 1 TO N
 90  READ M$ (I), S$ (I), A$ (I), T$ (I)
100  LET A (I) = LEN (A$ (I))
110  LET T (I) = LEN (T$ (I))
120  FOR J = A (I) + 1 TO 20
130  LET A$ (I) = A$ (I) + "  "
140  NEXT J
150  FOR K = T (I) + 1 TO 20
160  LET T$ (I) = T$ (I) + "  "
170  NEXT K
180  LET L$ (I) = M$ (I) + S$ (I) + A$ (I) + T$ (I)
190  NEXT I
1000 FOR J = 1 TO N – 1
1010 FOR K = J + 1 TO N
1020 IF L$ (J) < L$ (K) THEN GOTO 1060
1030 LET E$ = L$ (J)
1040 LET L$ (J) = L$ (K)
1050 LET L$ (K) = E$
1060 NEXT K
1070 NEXT J
1080 PRINT: PRINT: PRINT
1090 PRINT "BUSQUEDA POR MATERIAS PULSA 1"
1100 PRINT "BUSQUEDA POR AUTOR PULSA 2"
1110 PRINT "BUSQUEDA POR TITULO PULSA 3"
1120 INPUT "QUE OPCION DESEAS?"; 0
```

```
1130  IF O=1 THEN GOTO 1200
1140  IF O=2 THEN GOTO 1400
1150  IF O=3 THEN GOTO 1600
1160  IF O<>1 AND O<>2 AND O<>3 THEN GOTO
      1120
1170  GOTO 1090
1200  CLS
1210  INPUT "PON LOS DOS DIGITOS DE LA MATERIA
      QUE QUIERES VER"; X$
1220  FOR I=1 TO N
1230  IF X$=LEFT$ (L$ (I), 2) THEN GOTO 1250
1240  GOTO 1260
1250  PRINT L$ (I)
1260  NEXT I
1270  INPUT "QUIERES VOLVER AL MENU? (S/N)";
      Y$
1280  IF Y$="N" THEN STOP
1290  IF Y$<>"S" AND Y$<>"N" THEN GOTO 1270
1300  CLS
1310  GO TO 1080
1400  CLS
1410  INPUT " QUE AUTOR QUIERES VER?"; N$
1420  LET Q=LEN (N$)
1430  FOR W=Q+1 TO 20
```

Programas

```
1440  LET N$=N$+" "
1450  NEXT W
1460  FOR I=1 TO N
1470  IF MID$ (L$ (I), 4, 20) =N$ THEN GOTO 1490
1480  GO TO 1500
1490  PRINT L$ (I)
1500  NEXT I
```

```
1510  INPUT "QUIERES VOLVER AL MENU? (S/N)"; Y$
1520  IF Y$="N" THEN STOP
1530  IF Y$<>"S" AND Y$<>"N" THEN GOTO 1510
1540  CLS
1550  GOTO 1090
1600  CLS
1610  INPUT " ESCRIBE EL TITULO"; Z$
1620  LET P=LEN (Z$)
1630  FOR R=P+1 TO 20
1640  LET Z$=Z$+" "
1650  NEXT R
1660  FOR I=1 TO N
1670  IF Z$=RIGHT$ (L$ (I), 20) THEN GOTO 1690
1680  GOTO 1700
1690  PRINT L$ (I)
1700  NEXT I
```



Programas

```
1710  INPUT "QUIERES VOLVER AL MENU? (S/N)"; Y$
1720  IF Y$ = "N" THEN STOP
1730  IF Y$ <> "S" AND Y$ <> "N" THEN GOTO 1710
1740  CLS
1750  GOTO 1090
5000  DATA 3
5010  DATA 21,2,JUAN RAMON JIMENEZ, PLATERO Y
      YO
5020  DATA 21,2,G. GARCIA MARQUEZ, LA HOJARAS-
      CA
5030  DATA 08,5,MARTINEZ, MATEMATICAS E.G.B.
```

AQUÍ VAN LOS DATOS DE LOS LIBROS. AÑADE TODAS LAS SENTENCIAS DATA QUE NECESITES (Y NO OLVIDES MODIFICAR ADECUADAMENTE LA LÍNEA 5000)

NOTAS PARA EL ADULTO

Este es un programa-tipo, escrito en un BASIC convencional. Como ya conocen, será necesario introducir algunos cambios mínimos para poder ejecutarlo en determinados ordenadores. Deben tener en cuenta todas las observaciones efectuadas a lo largo de los capítulos anteriores a este respecto.

En especial, reiteramos la importancia de escribir adecuadamente las sentencias DATA, efectuar correctamente los dimensionados y modificar donde sea preciso aquellas instrucciones como LEFT$, MID$ y RIGHT$ que no existen en determinados ordenadores.

BIBLIOGRAFIA CONSULTADA

BELLIDO. *Cómo programar su SPECTRUM.*
Ed. PARANINFO, 1984. Madrid.

BELLIDO. *ZX81. Curso de programación BASIC.*
Ed. PARANINFO, 1984. Madrid.

BELLIDO y SANCHEZ. *BASIC para estudiantes.*
Ed. PARANINFO, 1987. Madrid.

BELLIDO y SANCHEZ. *BASIC para maestros.*
Ed. PARANINFO, 1984. Madrid.

CHECROUN. *BASIC. Programación de microordenadores.*
Ed. PARANINFO, 1984. Madrid.

GOMEZ, DEL PINO. *Curso completo de BASIC.*
T.I. Fundamentos. (2.ª Edic.). 1989.

GOMEZ, DEL PINO. *Curso completo de BASIC.*
T.II. Métodos. (2.ª Edic.). 1989.

GOMEZ, DEL PINO. *Curso completo de BASIC.*
T.III. Ficheros. 1988.

HART. *Diccionario del BASIC.* (2.ª Edic.). 1985.

LARRECHE. *BASIC. Iniciación a la programación.*
Ed. PARANINFO, 1984. Madrid.

RAMON. *44 Superprogramas en BASIC.* 1985.

ROSSI. *BASIC. Curso acelerado.* Ed. PARANINFO, 1985. Madrid.

SPENCER. *BASIC programming.* CAMELOT, 1983. New York.

VARIOS. *BASIC básico. Curso de programación.*
Ed. AGUADO, 1984. Madrid.

Notas

Notas

Diseño de la cubierta: Emil Tröger

Círculo de Lectores, S.A.
Valencia 344, 08009 Barcelona
1 3 5 7 9 0 9 0 3 8 6 4 2

Licencia editorial para Círculo de Lectores
por cortesía de Editorial Paraninfo, S.A.
Está prohibida la venta de este libro a personas que no pertenezcan a
Círculo de Lectores.

© Sofía Watt y Miguel Mangada
Madrid (España)

© Editorial Paraninfo, S.A.
Madrid (España)

Depósito legal: B. 5542-1990
Impresión y encuadernación: Printer industria gráfica, s.a.
N. II, Cuatro caminos s/n, 08620 Sant Vicenç dels Horts
Barcelona, 1989. Printed in Spain
ISBN: 84-226-3137-7
N.º 23770

Este libro responde a muchas cuestiones de la programación en BASIC. Y lo hace de una manera clara y sencilla, de forma que cualquier niño podrá —utilizando su ordenador— desde organizarse su propio fichero de direcciones hasta listar alfabéticamente una relación nominal, pasando por autoevaluar sus conocimientos escolares u organizar su propia liga de baloncesto...

Apenas son necesarios unos mínimos conocimientos de programación. A lo largo del texto, numerosas notas aclaratorias, comentarios y guías didácticas explican las dudas que puedan surgirle al niño y amplían sus conocimientos.

Utilizando ejemplos sencillos y directamente relacionados con su entorno, este libro confiere al niño un sólido conocimiento de la programación en BASIC.

23770